1986-06

Implementation of a material database system in Hellenic Armed Forces Formations

Bozikas, Panagiotis Andreou

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

IMPLEMENTATION OF A MATERIAL DATABASE SYSTEM IN HELLENIC ARMED FORCES FORMATIONS

by

Panagiotis Andreou Bozikas

June 1986

Thesis Advisor: L. Rawlinson

ECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

a REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
a SECURITY CLASSIFICATION AUTHORITY	3 DISTRIBUTION / AVAILABILITY OF REPORT Approved for public release; distribution is unlimited.
b DECLASSIFICATION / DOWNGRADING SCHEDULE	
PERFORMING ORGANIZATION REPORT NUMBER(S)	5 MONITORING ORGANIZATION REPORT NUMBER(S)
a. NAME OF PERFORMING ORGANIZATION Naval Postgraduate School 6b OFFICE SYMBOL (If applicable) 52	7a NAME OF MONITORING ORGANIZATION Naval Postgraduate School
c. ADDRESS (City, State, and ZIP Code) Monterey, CA 93943-5000	7b. ADDRESS (City, State, and ZIP Code) Monterey, CA 93943-5000
a NAME OF FUNDING / SPONSORING ORGANIZATION 8b OFFICE SYMBOL (If applicable)	9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
c. ADDRESS (City, State, and ZIP Code)	10 SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO / PROJECT NO / TASK NO / WORK UNIT ACCESSION NO

1 TITLE (Include Security Classification)
IMPLEMENTATION OF A MATERIAL DATABASE SYSTEM IN HELLENIC ARMED FORCES FORMATIONS

2 PERSONAL AUTHOR(S)
BOZIKAS, Panagiotis A.

13a TYPE OF REPORT	13b TIME COVERED	14 DATE OF REPORT (Year, Month, Day)	15 PAGE COUNT
Master's Thesis	FROM _____ TO _____	1986 June 20	159

6 SUPPLEMENTARY NOTATION

17 COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Database

19 ABSTRACT (Continue on reverse if necessary and identify by block number)

This thesis contains an implementation of a material database system for the Hellenic Armed Forces. The Hellenic Armed Forces Formations currently manage all material data manually. The author proposes the switching from manual processing to automated processing in an Ordnance Battalion on an Infantry Division using dBASE II with an IBM personal computer or compatible.

1. The reasons and the system task.
2. The conversation of the old manual system to an automated system.
3. The implementation of the design.

20 DISTRIBUTION / AVAILABILITY OF ABSTRACT ☐ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS	21 ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED
22a NAME OF RESPONSIBLE INDIVIDUAL CDR Linda Rawlinson,	22b TELEPHONE (Include Area Code) 408 646-2735 22c OFFICE SYMBOL 52RV

DD FORM 1473, 84 MAR 83 APR edition may be used until exhausted
All other editions are obsolete

SECURITY CLASSIFICATION OF THIS PAGE
UNCLASSIFIED

Approved for public release, distribution unlimited

Implementation of a Material Database System in Hellenic Armed Forces Formations

by

Panagiotis Bozikas
Lieutenant Colonel, Hellenic Army
B.A., Hellenic Army Academy, 1966

Submitted in partial fulfillment of the requirements
for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
June 1986

ABSTRACT

This thesis contains an implementation of a material database system for the Hellenic Armed Forces. The Hellenic Armed Forces Formations currently manage all material data manually. The author proposes the switching from manual processing to automated processing in an Ordnance Battalion of an Infantry Division using dBASE II with an IBM personal computer. Particular emphasis is placed on:

1. The reasons and the system tasks.
2. The conversion of the old manual system to an automated system.
3. The implementation of the design.

TABLE OF CONTENTS

LIST OF FIGURES

DEDICATION

In memory of my Father, Andreas Bozikas, who died on October 31, 85 in my country of Greece, during the period I was writing this thesis.

ACKNOWLEDGEMENT

Although my name appears on the cover and I wrote all of this thesis, I couldn't have done it without the help from a lot of other people whose names appear below. Their contributions range from essential help in analysis and design of the problem and use of dBASE to a simple conversation that sparked an idea. I am very grateful to everyone who freely shared their dBASE experiences and knowledge with me.

So, first of all and primarily, I would like to express my deep gratitude to my thesis advisor CDR L. C. Rawlinson of the Department of Computer Science, for her help to me by reading the thesis and suggesting improvements and several corrections.

Secondly, I thank my second reader, CDR S. Gary Baker of the Department of Computer Science, for his help to me, reading and making some corrections to my thesis.

Finally, I would like to express my thanks to my wife Roula, especially for her efforts in giving me encouragement and support when difficulties appeared during the writing of my thesis. Also, I would stress the moral support of my son Andy and my daughter Gina. They gave me the necessary time to complete my thesis.

To the above people I am very grateful because they freely shared their experiences, knowledge, and friendship with me to finish my thesis.

I. GENERAL DATABASE CONCEPTS

A. INTRODUCTION

A database is a shared collection of inter-related data designed to meet the varied information needs of an organization. In the early 1970s, database processing was considered an esoteric subject, of interest only to the largest corporations with the largest computers. Today database processing is becoming an information systems standard. In some computers all data is organized into a database, while files, as individual entities, do not exist.

The change to database processing has occurred largely because of economics. Database processing favors people at the expense of computers. Programmers and other users of data can be more efficient and effective using database processing, accomplishing more within a fixed amount of time. On the other hand, database processing requires more computer resources than traditional file processing does.

In the last ten years the cost of labor has been increasing steadily. Even worse, management has encountered great difficulty in acquiring and keeping competent systems development personnel. Meanwhile, the cost of computers has decreased dramatically. Simply stated, people have become more expensive while machines have become cheaper. In parallel, database processing has offered the potential for trading people resources for machine resources. The result has been a substantial increase in the number of database applications. By all standards, these trends are likely to continue. Conceivably, files as independent entities might

disappear entirely, perhaps in another ten years all commercial data processing will become database processing.

1. Traditional File Processing Approach

Database technology allows an organization's data to be processed as an integrated whole. It reduces artificiality imposed by individual files for separate applications and permits users to access data more naturally. With a file processing approach, each application program maintains its own files. There is no sharing of data among different application programs.

Consider the three systems shown in Figure 1.1. These are file processing systems, they are predecessors of database systems. With file processing, each file is considered to exist independently. The payroll system in Figure 1.1 processes only the faculty data file, the class scheduling system processes only class data, and the grade posting system processes only student data. These systems are effective in that they produce checks, schedule classes, and record grades. [Ref.1: pp. 1-3]

2. Database Processing Approach

Suppose it is necessary to know the salary paid to each instructor who teaches a class scheduled by the class scheduling system in Figure 1.1. To obtain this information, a new program must be written to extract data from both the faculty and the class data files. Unfortunately, there is no guarantee that these files are compatible. The faculty data file might be written in COBOL binary format, whereas an incompatible PL/I record format might be used for the class

data file. If so, one file must be converted to the format of the other, and then the extraction program written, tested, and run. This process will take time. In some cases, conversion entails so much effort to eliminate incompatibility that it simply cannot be done within a reasonable cost.

Figure 1.2 shows a database processing system. The files in Figure 1.1 have been integrated into a database that is processed indirectly by the application programs. The payroll, class scheduling, and grade posting systems can perform their old functions, but the programs call upon the database management system (DBMS) to access the database. The DBMS is a complex and usually large program that acts as a data librarian. It stores and retrieves data. In order to perform its functions, the DBMS stores not only data, but also a description of the format of the data.

The faculty, class, and student data in Figure 1.2 can be processed as an integrated whole. Since the files have been created by the DBMS, all of the data is compatible. Further, the DBMS may have features to enhance integrated processing. For example, a faculty record can be logically 'tied to' several class records to represent the relationship between teacher and class. Thus database processing is integrated processing. [Ref.1: p.3]

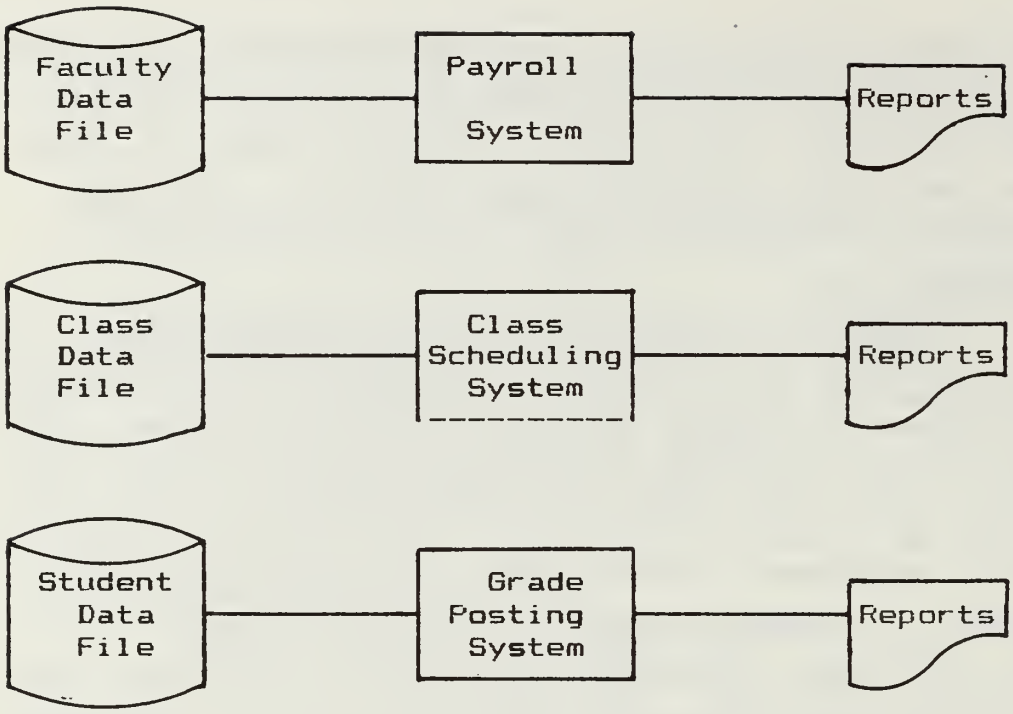


Figure 1.1 : Three File Processing Systems [Ref.1: p.2].

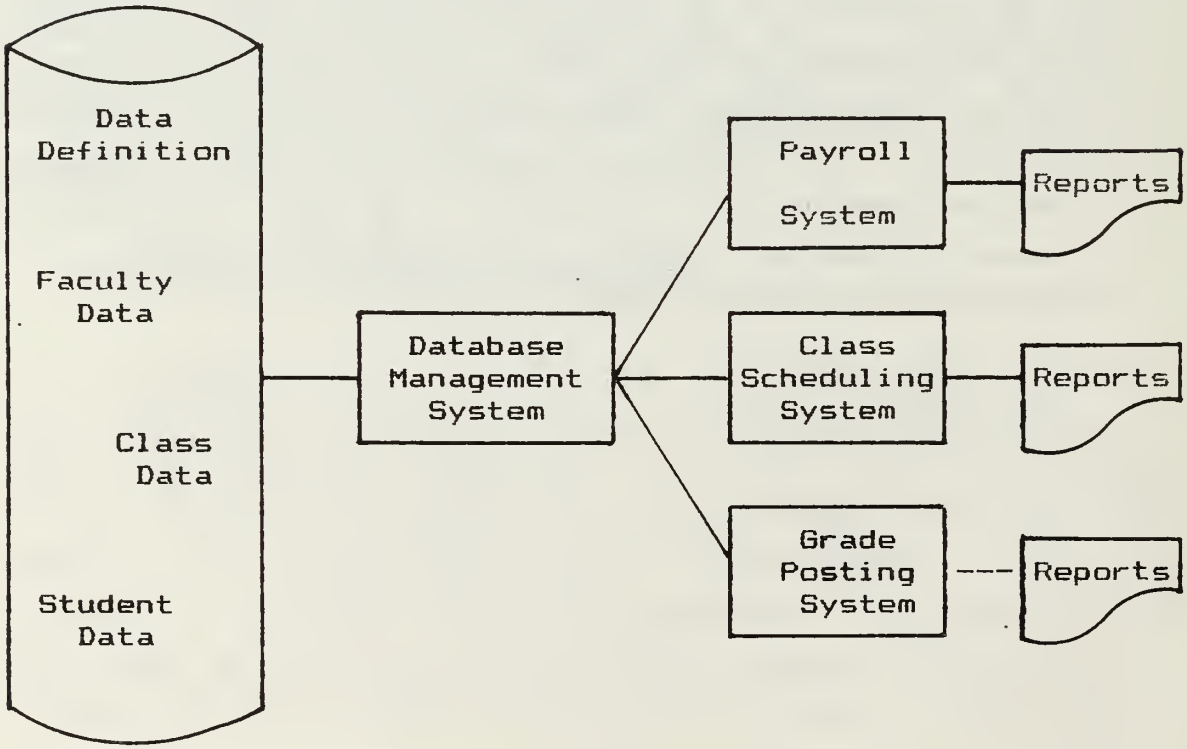


Figure 1.2 : A Database Processing System [Ref.1: p.4].

3. Comparison of Traditional and Database Approaches

The table below shows the differences between the Tradition file processing approach and the Database processing approach, comparing the characteristics of each:

Tradition processing approach	Database processing approach
Uncontrolled Redundancy Each application has its own files, i.e. same information could be repeated across different files.	**Minimal data Redundancy** Previously separated and redundant data files are integrated into a single, logical structure.
Inconsistent data Because of redundancy, data may become inconsistent, e.g. if there is a change in any file, say instructor pay,then the change must also be in all other files that contain this information.	**Consistent data** Because the redundancy is controlled, there is a less chance of inconsistency.
Limited data sharing Since every application has its own private files, there is little opportunity to share data between application program files.	**Sharing of data** Data can be shared by many application programs through the DBMS.
Poor enforcement of standards A standard in data formats is very hard to maintain if each application program has its own files.	**Enforcements of standards** Since data is stored only once, maintaining a standard is a lot easier.
Excessive program maintenance Because the descriptions of files, records, and data fields are embedded within individual application programs, any change in a data file will necessitate a change in the program.	**Reduced program maintenance** Due to data independence.

[Ref. 1: pp.3-8]

4. Components of a Database System

A database system is a collection of five components that interact to satisfy user needs. The five components are hardware, programs, data, people, and procedures.

a. Hardware

Database systems do not require any special type of hardware. Database applications often require more hardware, more main memory, a faster CPU, and more direct access storage.

As shown in Figure 1.3, the computer processing the application program sends requests for service and data over a channel to the database machine. The machine processes the requests and sends results, data, or messages back to the main computer. Thus database processing can be performed simultaneously with applications processing. [Ref. 1: pp. 8 & 9]

b. Programs

Several types of programs are used in database processing systems. Figure 1.4 shows the approximate relationships of the major types. Online processing requests or transactions are provided by users at terminals. The requests are sent to the processing computer over communications lines. The requests are received and routed by the communications control program (CCP). This program has several important functions. It provides communications error checking and correction, it coordinates terminal activity, it routes messages to the correct next destination,

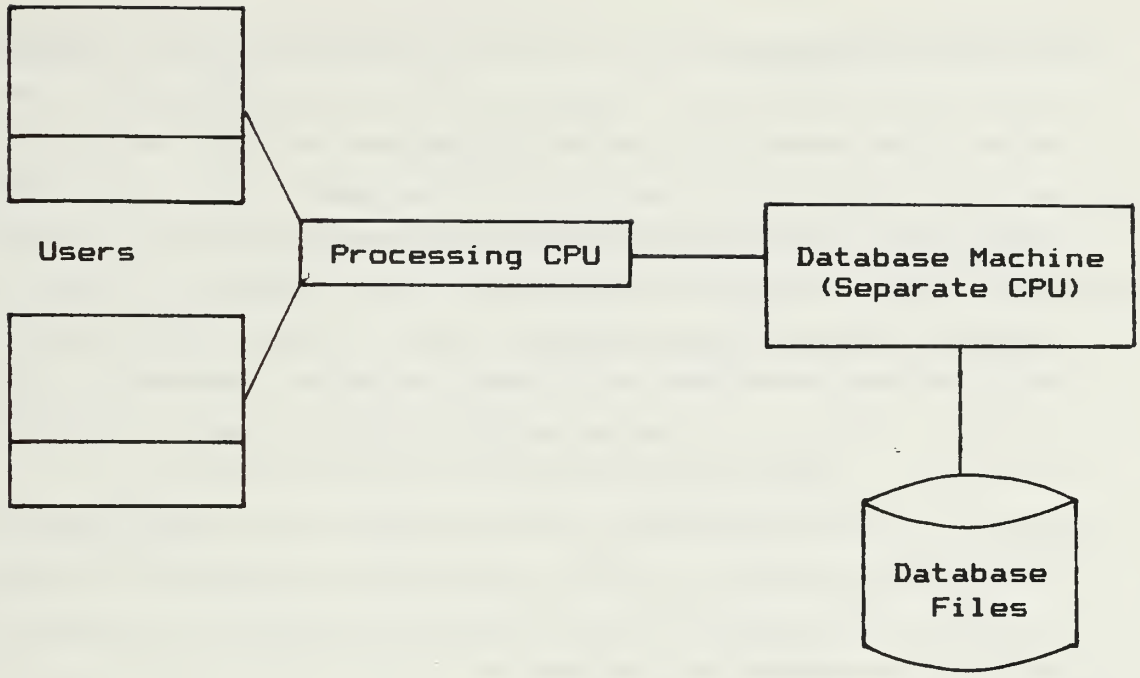


Figure 1.3: Processing with Database Machine [Ref.1: p.8].

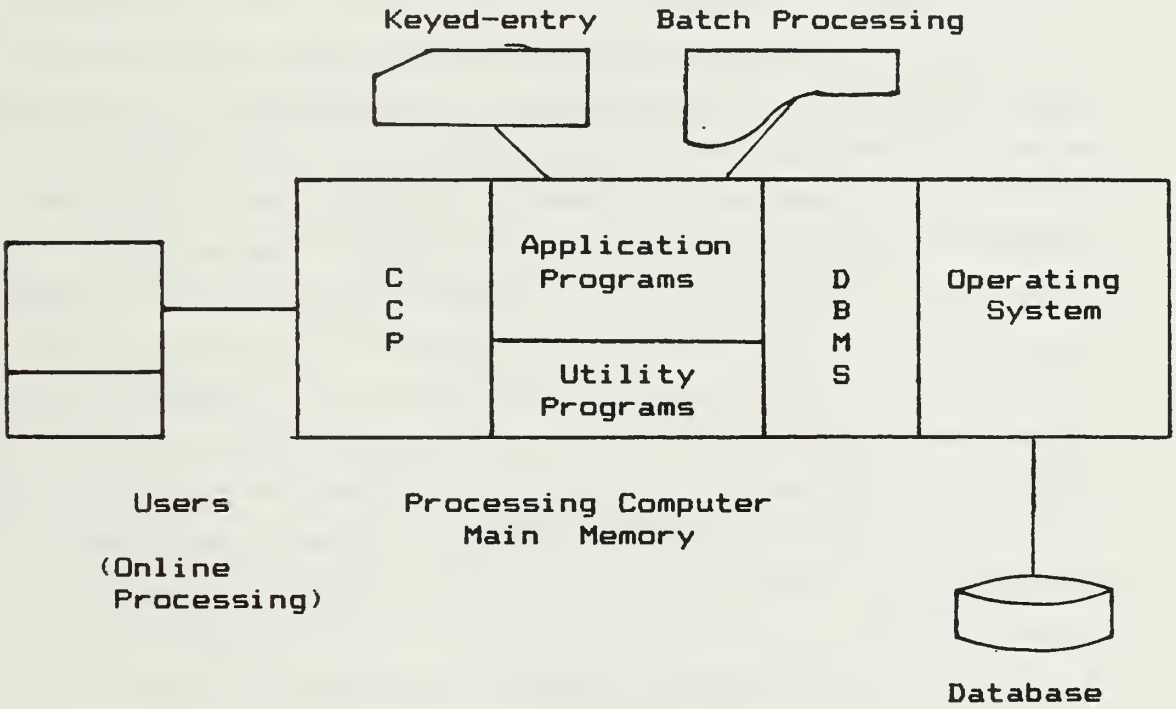


Figure 1.4: Programs Involved in Typical Database Processing [Ref.1: p.9].

it formats messages for various types of terminal equipment, and it performs other communication-oriented tasks. The CCP is an important and complex program. The CCP routes online input to the next level of the programs. This level contains application programs (AP) as well as database utilities. The application programs satisfy specific needs like order entry, inventory accounting, billing, and so forth. The utility programs are provided by either the DBMS or the hardware vendor. These programs provide a wide variety of services.

DBMS is a software system that carries out all user's requests for data. The request may be an update or a retrieval operation. The final type of program involved in database processing is the operating system. This set of programs manages the computer's resources. [Ref.1: pp 9-11]

c. Data

A database is a self-describing collection of intergated files. The database is self-describing because it contains, within itself, a description of its structure.

According to standard usage in the computer industry, bits are grouped into bytes or characters, characters are grouped into fields, and fields are grouped into records. A collection of records is called a file. It is tempting to continue this abstraction by stating that files can be grouped to form a database, which is more than just a collection of files, but a collection of integrated files. Another way of saying this is that a database is a collection of files and the relationships among records in those files.

In a database system a variety of forms, or views,of the data are defined. One such view is called the

schema, or conceptual view. This is the complete, logical view of the data.

The schema describes all of the data in the database. For a bank, the database schema might include the customer, checking, savings, loan, and credit records.

It would be undesirable from a control standpoint to allow every application program to access all of this data. The bank would not want the checking programs, for example, to have access to loan data.

To restrict access to the database, companies define another type of view called a subschema, or external view. The subschema defines a subset of the schema to be seen by a given application program or user.

Figure 1.6 is an example of subschemas for a Bank database. A program that invokes the checking subschema will see only customer and checking records. All other database records will be invisible to that program. There will be more than one subschema and subschemas can overlap.

A third view of the data is called the internal or the physical view. This is the form of the data as it appears to a particular processing computer. It describes how data is physically arranged and how it is allocated to files. [Ref.1: pp. 11-14]

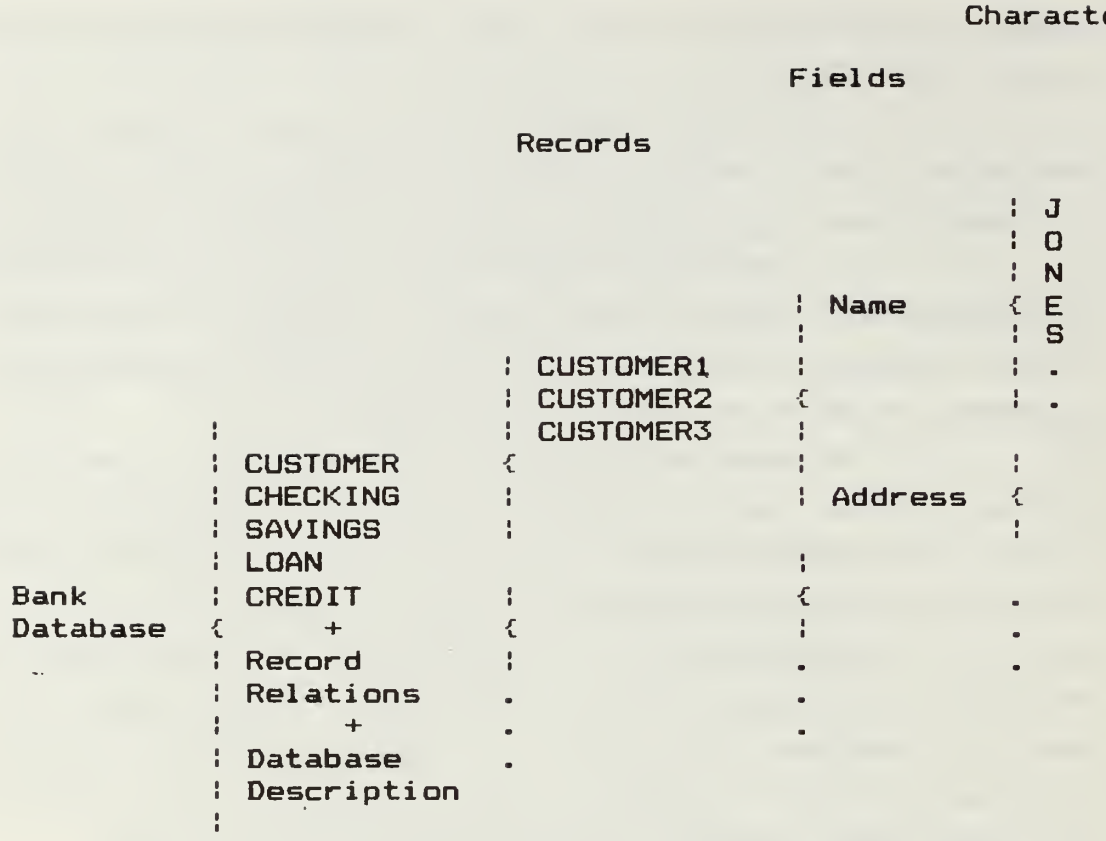


Figure 1.5: Composition of Bank Database.[Ref.1: p.13]

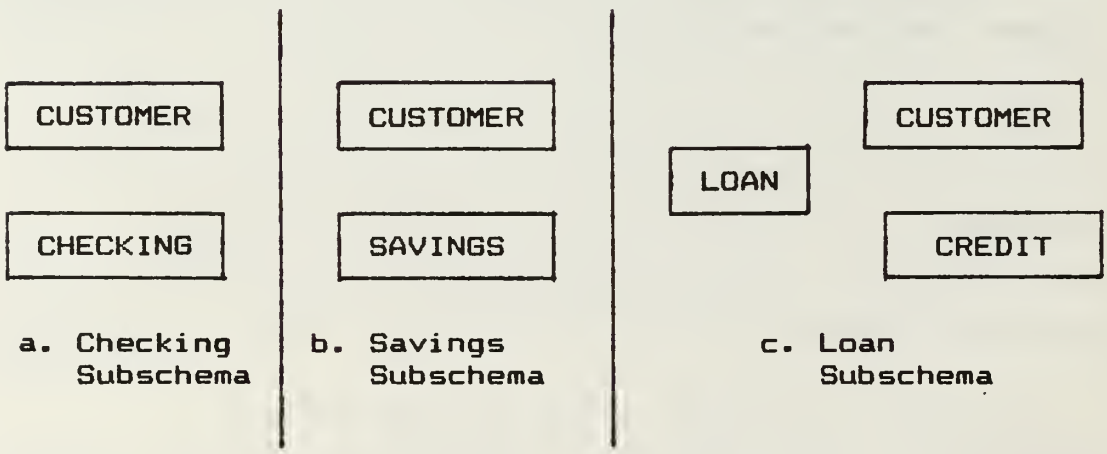


Figure 1.6: Subschema for a Bank Database.[Ref.1: p.14]

d. People

People are the fourth component of a business database system. These consist of many different types:

(1) Clientele: They are the people for whom the system is developed. The clientele of a payroll system are employees.

(2) Users: They are the people who employ the system to satisfy a particular need.

(3) Operations personnel: They run the computer and associated equipment.

(4) Systems development personnel: They design and implement the database system. They determine requirements, specify alternatives, design the five components of the system, and manage systems implementation.

(5) Database administration (DBA) personnel: The function of the DBA staff is to serve as a protector of the database and as a focal point for resolving user's conflicts. The DBA should be a representative of the community as a whole, and not of any particular user or group of users.

e. Procedures

Both users and the operations staff need documented procedures for normal conditions. The users need to know how to sign on to the system, how to use the terminals, how to provide data, and so forth. They also need procedures that ensure they do not interfere with one another.

Computer operations personnel also need procedures. They need to know how to start and stop the database applications, and how to perform backup. Further, if there are special operating instructions such as for mounting certain disks or the like, these instructions need to be documented as standard procedures. [Ref.1: pp.15-17]

B. DATA MODELS

The study of database processing involves learning the database models. In this section the data models are presented.

A model is a representation of real-world objects, events and their associations. A data model is an abstract representation of the data about entities, events, activities, and their associations.

The purposes of data models are:

1. To represent data.

2. To be understandable.

1. Components of a database model

Database models have two major components. The data definition language (DDL) is a vocabulary for defining the structure of the database. The DDL must include terms for defining records, fields, keys, and relationships. In addition, the DDL should provide a facility for expressing a variety of user views. Ideally, the model will also provide a method for expressing database constraints. The purpose of data definition components in a database model are to:

a. Describe records.
b. Describe fields.
c. Describe relationships.
d. Identify keys.
e. Express user views.
f. Express constraints:
    (1) Edit.

(2) Intrarecord.

(3) Intrerrecord.

Data manipulation language (DML) is the second component of a database model. The DML is a vocabulary for describing the processing of the database. Facilities are needed to retrieve and change data in the database. Two types of DML exist.

Procedural DML is a language for describing actions to be performed on the database. Procedural DML obtains a desired result by specifying operations to be performed. For procedural DML, facilities are needed to define the data to be operated on and to express the actions to be taken. Both data items and relationships can be accessed or modified. Also, to ensure that the database can be accurately recovered in the case of failure, commands are needed to define logical transactions and to eliminate changes in case of a program-detected error.

Nonprocedural DML is a language for describing the data that is wanted without describing how to obtain it. The DBMS is given the job of determining how to get the result. Nonprocedural DML is descriptive, not prescriptive.

2. Hierarchical Data Model (HDM)

A model is hierarchical if its only data structure is a hierarchy (tree). HDM represents data as a set of nested one-to-many (1:M) and one-to-one (1:1) relationships.

One obvious problem with HDM is that many-to-one (M:1) relationship is not supported, thereby producing redundancy. Suppose one teacher teaches many courses. That teacher's information will be repeated for every course he teaches. The advantage of HDM's is that they are familiar structures.

A hierarchical database consists of one or more trees with each tree consisting of an hierarchy of records as shown in Figure 1.7.

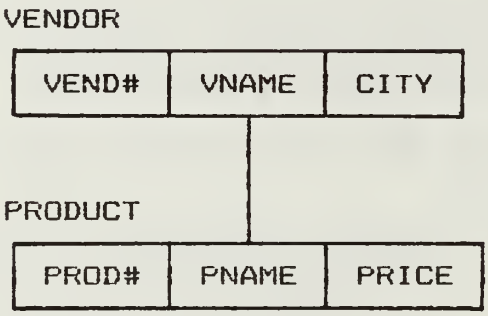


Figure 1.7: Hierarchical Diagram For Database.

The hierarchy of Figure 1.7 is declared as follows:

```
TREE NAME IS VENDTREE
 RECORD  NAME  IS  VENDOR
      VEND# TYPE IS INTEGER
      VNAME TYPE IS CHAR(20)
      CITY  TYPE IS CHAR(10)
 RECORD  NAME  IS  PRODUCT
      PARENT IS VENDOR
      PROD# TYPE IS INTEGER
      PNAME TYPE IS CHAR(30)
      PRICE TYPE IS REAL
```

The parent/child relationship is denoted by putting **PARENT IS <record name>** in the child record declaration.

a. Data Retrieval

The basic retrieval command is **GET** and the general format is: **GET LEFTMOST <target record name> [WHERE <condition>]**

The <condition> has the following form:

<record name>.<field name> <rel-op><value>

where <rel-op> is usually a binary relational operator. The <record name> must be an ancestor of <target record name>. The square brackets signify the optional part.

For example, referring to Figure 1.7, we want the price of the PRODUCT whose name is AAA and belongs to the 1st VENDOR.

```
GET LEFTMOST PRODUCT
    WHERE (VENDOR.VEND# = 1)
    AND (PRODUCT.PNAME = 'AAA'
PRINT PRODUCT.PRICE
```

(1) Insertion. Inserting a new record is very simple. First put the values in the template, (Template is a kind of data. Values placed in the template will be copied to the database by STORE command and copied into it from the database by the GET command) and then execute the INSERT command, which has the same format as the GET command. To add a new PRODUCT 'AAA' to the 1st VENDOR, we write:

```
PRODUCT.PNAME = 'AAA'
PRODUCT.PROD# = '100'
INSERT PRODUCT
   WHERE VENDOR.VEND# = 1
```

So the format of insert command is:

INSERT <record name>

(2) Deletion. To delete a record, we first locate the desired record by using the GET HOLD command and then we write DELETE. GET HOLD works exactly like the GET command. The GET HOLD will make sure that no other pointers are pointing to the descendant of the record to be deleted. When a record is deleted, all descendent records of the record will also be deleted automatically. So to delete the 2nd PRODUCT of the 4th VENDOR, we would simply state:

```
GET HOLD LEFTMOST PRODUCT
   WHERE (VENDOR.VEND# = 4)
   AND (PRODUCT.PROD# = 2)
DELETE
```

So the format of delete command is:

```
GET  HOLD <record name>
DELETE
```

(3) Modification. The modification operation is similar to deletion. To modify a record after gaining exclusive control of it, we first change the data values in the template and then issue the command **REPLACE**. The following will replace the PRICE of the 7th PRODUCT to $40.

```
GET HOLD LEFTMOST PRODUCT
   WHERE PRODUCT.PROD# = 7
PRODUCT.PRICE = '$40'
REPLACE
```

[Ref.2: pp.30-32]

3. Network Data Model (CODASYL DBTG)

The Network Data Model (NDM) represents data as a set of record types and pairwise relationships between record types. Relationships that involve more than two record types are not directly permitted. NDM does not allow M:N (Many to Many) relationships. It supports the use of multiple 1:M (One to Many) relationships between the same pair of record types.

The CODASYL DBTG data model was developed by the same group that formulated COBOL. It was developed during the late 1960s and is the oldest of the data models. The DBTG model is a physical database model. There are constructs for

defining physical characteristics of data, for describing where data should be located, and for instructing the DBMS regarding what data structures to use for implementing record relationships and other similar physical characteristics.

A procedural DML has been carefully and completely defined for this model. A nonprocedural DML for the DBTG model does not exist.

Many DBMS products are based on the CODASYL DBTG model. Consider the following data structure diagram.

Figure 1.8: Network Diagram For Database.

The above diagram shows a one to many (1:M) relationship between VENDOR and PRODUCT. In a network DBMS, we declare them (using DDL) as in Figure 1.9

```
RECORD  NAME  IS  PRODUCT
   LOCATION MODE IS CALC USING VEND
   1 PROD# TYPE IS INTEGER
   1 PNAME TYPE IS CHAR(20)
   1 UNITPRICE TYPE IS REAL
RECORD  NAME  IS  VENDOR
   LOCATION MODE IS VIA SHIPMENT
   1 VEND# TYPE IS INTEGER
   1 VNAME TYPE IS CHAR(30
   1 ADDR TYPE IS CHAR(10)
   1 OWNER TYPE IS CHAR(20)
SET  NAME  IS  SHIPMENT
   OWNER IS VENDOR
   MEMBER IS PRODUCT
        INSERTION IS AUTOMATIC
        RETENTION IS FIXED
        SET SELECTION IS VALUE OF PROD#
```

Figure 1.9: A DDL Description of Records and Set Types.

In Figure 1.9 we see the declaration of the record types and set types needed to describe the network of Figure 1.8. These declarations are in DDL syntax.

The fields for each record type, with the data type of each field, are listed in the declaration of the record type. The integer 1 preceding each field name is a level number, as in the declaration of PL/I record structures. Level numbers up to 99 are permitted, allowing fields to have structure. The typical use of such structure is to declare, within a field such as ADDRESS, subfields like STREET, CITY, and ZIP CODE at level 2.

So a record type is declared as:

RECORD NAME IS <record type name>

followed by the list of attribute names and their data type.

Figure 1.9 also contains the declaration of one, set type, corresponding to the one link in Figure 1.8. The link is many-to-one from PRODUCT record type to the VENDOR record type. A set type is declared as follows:

SET NAME IS <set type name>

followed by the name of OWNER and MEMBER record types.

The meaning of an option available in declaring a record type, LOCATION MODE IS, is explained as follow:

CALC: Indicates that a record instance will be placed and may be accessed in secondary memory based on a value for a primary or secondary key. Usually, this is implemented by key value hashing, but index methods are possible. That is, a database can be entered directly at a given record if a CALC key value is known.

VIA : Indicates that a record instance will be placed in secondary memory close to its parent record instance for one specified set. This helps to improve performance when used with a frequently referenced set. VIA and CALC may not both be used on the same record.

Use of VIA prevents access to the record by a key value.

DIRECT: Indicates that a record instance will be placed by and may be accessed by its physical address.

a. Program Environment

For every user accessing a database, there is a workspace called user working area which holds three kinds of data:

(1) Variables. They are defined by the program.

(2) Currency pointers. They are variables that hold the physical address (called database key in CODASYL terminology) of the record instance most recently accessed or manipulated in a specified category of records. There are three kinds :

   (a) **Current of run-unit:** The most recent instance of any database record is referenced.

   (b) **Current of record type:** For each record type in the schema, the most recent instance is referenced.

   (c) **Current of set type:** For each set type in the schema, the most recent set record instance (either owner or member) is referenced.

(3) Templates for the various record types. Values placed in the template will be copied to the database by STORE command and copied into it from the database by the GET command.

b. Data Retrieval

Two key commands in data relationnal operations are:

(1) FIND. The FIND command moves the currency pointers. It has the following syntax:

(a) FIND <record type> RECORD BY DATABASE.
For example, referring to Figure 1.8 and assuming:
- VENDOR is stored with LOCATION MODE BY CALC using VEND#.
- PRODUCT is stored VIA VENDOR

```
LOC = CURRENT OF PRODUCT
FIND PRODUCT RECORD BY DATABASE KEY LOC
GET PRODUCT, UNITPRICE
```

(b) FIND <record type>
For example:

```
VENDOR.VNAME = 'AAA'
FIND VENDOR RECORD BY CALC
GET VENDOR
```

(2) GET. This command copies the record instance of the current run-unit into a template, and has the following syntax:
GET <record type>,<list of attribute names>
Notice that a record instance of<record type> must be pointed by the current run-unit.

c. Data Update

An update operation includes insertion, deletion and modification of a record. There are several options available in network DBMS for the update operations. When we insert a new record into a database, do we want the record to be automatically attached to the appropriate owner or do we want to control where it is attached ourselves? Two options are available, INSERTION IS AUTOMATIC and INSERTION IS MANUAL. We would use INSERTION IS AUTOMATIC if the record

cannot exist without having an owner. On the other hand, if the record can exist without having an owner, then we must select INSERTION IS MANUAL. The option INSERTION IS affects the insertion operation. Similarly, the RETENTION IS option affects the removal of a member record from a set. When **RETENTION IS FIXED** is used, the member record must be permanently associated to the same owner. The only way to change the association is to physically erase the record from the database, then re-insert it connecting to the new owner. When **RETENTION IS MANDATORY** is used, the member record does not have to be fixed to the same owner. However, the member record must be associated to some owner in order to exist in the database. When **RETENTION IS OPTIONAL** is used, the member record does not have to be fixed to the same owner nor associated to some owner. **[Ref.2: pp. 25-30]**

4. Relational Data Model (RDM)

The relational data model is near the midpoint between human-machine activity because it has both logical and physical characteristics. The relational model is logical in that data is represented in a format familiar to humans. The relational model is unconcerned with how the data is represented in computer files. There are relational DBMS products. This means that databases designed according to the relational model need not be transformed into some other format before implementation.

RDM is different from HDM and NDM in its architecture and also in:

a. Normalization theory.
Properties of a database that make it free of certain maintenance problems.

b. Query languages.
These languages permit data to be manipulated as groups and not procedurally as one record at a time.

RDM represents data as a collection of relations. A relation is a 2-dimensional table with the following properties:

a. Each column contains values about the same attribute.

b. Each column has a distinct name.

c. Each row is distinct.

d. Sequence of the rows is immaterial.

Figure 1.10 shows two relations, one for customers and the other for invoices. The rows of the relation are the file records. Rows are sometimes called tuples of the relation. The fields of the relation are shown in the columns, they are sometimes called the attributes of the relation.

The significance of the relational model is not that data is arranged in relations, but that relationships are considered to be implied by data values. For example, in Figure 1.10 a relationship is implied by the common attribute, customer-number. Customer-number 10 in the CUSTOMER relation is related to 100 in the INVOICE relation. We can join these two relations together on a common value of Customer-numbers as shown in Figure 1.11.

The principal advantage of carrying relationships in data is flexibility. Relationships need not be predefined. We can join CUSTOMER tuples with INVOICE tuples or CUSTOMER tuples with ACCOUNTS-RECEIVABLE tuples, or whatever, without having to predefine the relationships in the design.

Customer number	Customer name	Other Data
10	A	
20	B	
30	C	

a. Customer Relation

Invoice number	Customer number	Other Data
100	10	
120	30	
140	20	
160	20	
180	30	
200	10	
220	10	

b. INVOICE Relation

Figure 1.10: Sample Relations. [Ref.1: p. 196]

Customer number	Customer name	Other Customer Data	Invoice number	Customer number	Other Invoice Data
10	A		100	10	
30	C		120	30	
20	B		140	20	
20	B		160	20	
30	C		180	30	
10	A		200	10	
10	A		220	10	

Figure 1.11: Join of CUSTOMER and INVOICE Relations.
[Ref. 1: p. 197]

More research has been done regarding the relational model than any other model. Much has been said regarding the best ways of structuring relations, and many different relational data manipulation languages have been designed.

Some of these, such as SQL, are nonprocedural. Others, such as relational algebra, are procedural. Until recently, the problem with the relational model has been the lack of a commercially viable relational DBMS. However, there are now several relational DBMS products.

a. Data Definition Language (DDL)

The general format of DDL in a relational DBMS is: CREATE <table name> (<attribute list>) where <table name> is the name of relation and <attribute list> is the list of attributes and their type declaration. To create an employee table with name, age, and address attributes, we declare.

```
CREATE EMPLOYEE (NAME=CHAR(20), AGE=INTEGER,ADDR=CHAR(10))
```

b. Data retrieval

The relational model is based on the mathematical theory of set and relations. We have well-defined mathematical formalisms for manipulating relations. They are:

(1) Relational algebra. and

(2) Relational calculus, divided into tuple and domain calculus.

Real query languages used in relational DBMS are:

(1) SQL (Sequence Query Language), based on relational algebra.

(2) QUEL (Query Language), based on tuple calculus.

The general formats are:

(1) SQL

```
SELECT <attribute>
FROM <relation>
WHERE <conditions>
```

(2) QUEL

```
{RANGE OF <var> IS <relation>}
RETRIEVAL (<var>.<attribute>
                    { , <var>.<attribute>})
WHERE <condition>
```

c. Data update

(1) Insertion. The general formats for the insertion operation are:

(a) SQL

```
INSERT  INTO <relation>
     [WHERE <condition>]
```

(b) QUEL

```
APPEND [TO] <relation>.(<target list>)
```

(2) Delete. The general formats for deletion operation are:

(a) SQL

```
DELETE  FROM  <relation>
     [WHERE <condition>]
```

(b) QUEL

```
DELETE <tuple var>
      [WHERE <condition>]
```

(3) Modification. They are:

(a) SQL

```
UPDATE <relation>
SET <assignment>
    [WHERE <condition>]
```

(b) QUEL

```
REPLACE <tuple var> (<target list>)
    [WHERE <condition>]
```

[Ref. 2: pp.19-25]

5. Comparison of the Models

To evaluate the three models we must first state the criteria by which they should be judged. We see two primary concerns.

a. Ease of use

Especially in small databases, on the order of thousands or tens of thousands of records, the principal cost may be the time spent by the programmer writing application programs and by the user posing queries.

b. Efficiency of Implementation

When databases are large, the cost of storage space and computer time dominate the total cost of implementing a database. We need a data model in which it is easy for the DBM to translate a specification of the conceptual schema and the conceptual-to-physical mapping into an implementation that is space efficient and in which queries can be answered efficiently.

By the criterion of ease of use, there is no doubt that the relational model is superior. It provides only one concept, the relation, that the programmers or user must understand. Moreover, the relational algebra and calculus clearly provide a notation that is quite powerful. The network model requires our understanding of both record types and links, and their interrelationships. The implementation of many-to-many relationships and relationships on three or more entity sets is not straightforward. Similarly, the hierarchical model

requires understanding the use of pointers and has the same problems as the network model regarding the representation of relationships that are more complex than many-to-one relationships between two entity sets.

When we consider the potential for efficient implementation, the network and hierarchical models score high marks. Since relations can, and often do, represent many-to-many mappings, we see that efficient implementation can be more difficult for relational than for network or hierarchical Database.

Early commercial database systems were almost uniformly based on the network or hierarchical model, because the emphasis of such systems has been on the maintenance of large databases, and these models lend themselves most easily to the necessary efficient implementation. However, there were in 1982 several successful commercializations of the relational model, and we feel that the relational systems will become progressively more accepted for two reasons. First, it is becoming clear that the same concepts used to design a large database apply as well to small and medium scale databases, and there are many more small databases than large ones. With small databases, the ease of use inherent in the relational model assumes increased importance. Second, many of the apparent inefficiencies of the relational model can be eliminated. [Ref.2: pp.168-170]

C. DBASE II CONCEPTS

DBASE is a relational database management system. The dBASE II does contain its own programming language, permitting a user to develop extremely powerful and complex programs that meet demanding applications like general personnel, accounting and inventory control.

1. Features of dBASE II

The most important features of dBASE II are:

a. Independence of programs and data. Changes in file structures do not affect programs.

b. Data can be easily updated.

c. Sorting and indexing capabilities.

d. Easy creation of reports by the report generator facility, or under program control.

e. Very high level built-in language which supports structured programming.

2. Limitations of dBASE II

The limitations of dBASE II are:

a. DBASE II allows only two files to be open at a time. This creates difficulties which can be overpassed by using special techniques but the system will slow down.

b. DBASE II allows only 32 fields per record, which is enough most applications, and the maximum number of characters permitted per record is 1000.

c. Each field can be up to 254 characters long.

d. DBASE II does not allow for swapping of data disks during a program execution, therefore making the selection of a floppy-disk based system impractical for database sizes > 1.2 Mbytes.

e. DBASE II allows 16 programs to be run at any given time, reduced by the number of data files in use. For example, if we have 2 files in use. then 14 programs are allowed.

f. DBASE II applications are slower than compiled programs.

D. PLANNING PHASE

Planning phase can generally be thought of as having two major components, systems analysis and systems design.

System analysis is the process of gathering and interpreting facts, diagnosing problems, and using the facts to improve the system.

System design is the process of planning a new system to replace or complement the old. System design proceeds through the two phases of logical design and physical design. **Logical design** includes, in detail, the specifications of the new system, that is, the outputs, the inputs, the files, and the system functions. **Physical design** includes the program software, and the working system.

System analysis and design are covered in the next three sections 2, 3, and 4.

II. ANALYSIS PHASE

System analysis is the process of gathering and interpreting facts, diagnosing problems, and using the facts to improve the system. Analysis specifies what the system should do.

In our case, we want to 'computerize' an Ordnance Battalion in order to better control the inventory and have more current, up-to-date, information about stock levels and reordering.

Before the design of the system can be determined for activities such as the capture of data, updating of files, and production of reports, we need to know more about how the Battalion now handles its operations. Especially, we must answer all the following questions. What forms are being used to store information manually such as requisitions and orders? What reports, if any, are now being produced and what they are being used for? How does the system presently work, or more specifically, what does the flow of information through the system look like? Why does the Battalion want to change its current operations? Does it have problems in handling inventory, orders, and supplying the units of the Division?

After answering the above questions and gathering other information about the current system we can begin to determine how and where a computer information system can benefit the user of the system.

This chapter contains a discussion of the current logistic, tactical, and operational system of the Ordnance Battalion. That is followed by the analysis which contains the objectives of this project, the automation environment, and the system requirements including input and output information.

A. CURRENT SYSTEM

1. General Description of an Ordnance Battalion

An Ordnance Battalion is the primary materials supplier to all units of an INFANTRY DIVISION. In total, it supplies materials for up to 25 independent units.

The organization of a typical Infantry Division and an Ordance Battalion is shown in Figures 2.1, and 2.2, respectively. Figure 2.3 shows a typical list of units supplied by the Ordnance Battalion in an Infantry Division.

2. Ordnance Battalion' Materials

An Ordnance Battalion handles two main categories of materials:

a. General Material.

(1) Transportation means (Vehicles, etc.)

(2) Communication means (Wireless sets, etc)

(3) Weapons (Rifles, Machine guns, Pistols, etc)

(4) Other general-material (Beds, Tables, Dresses,etc)

b. Spare-Parts.
Spare-Parts for the above general materials' list. Especially, spare-parts for transportation means, communication means, and weapons.

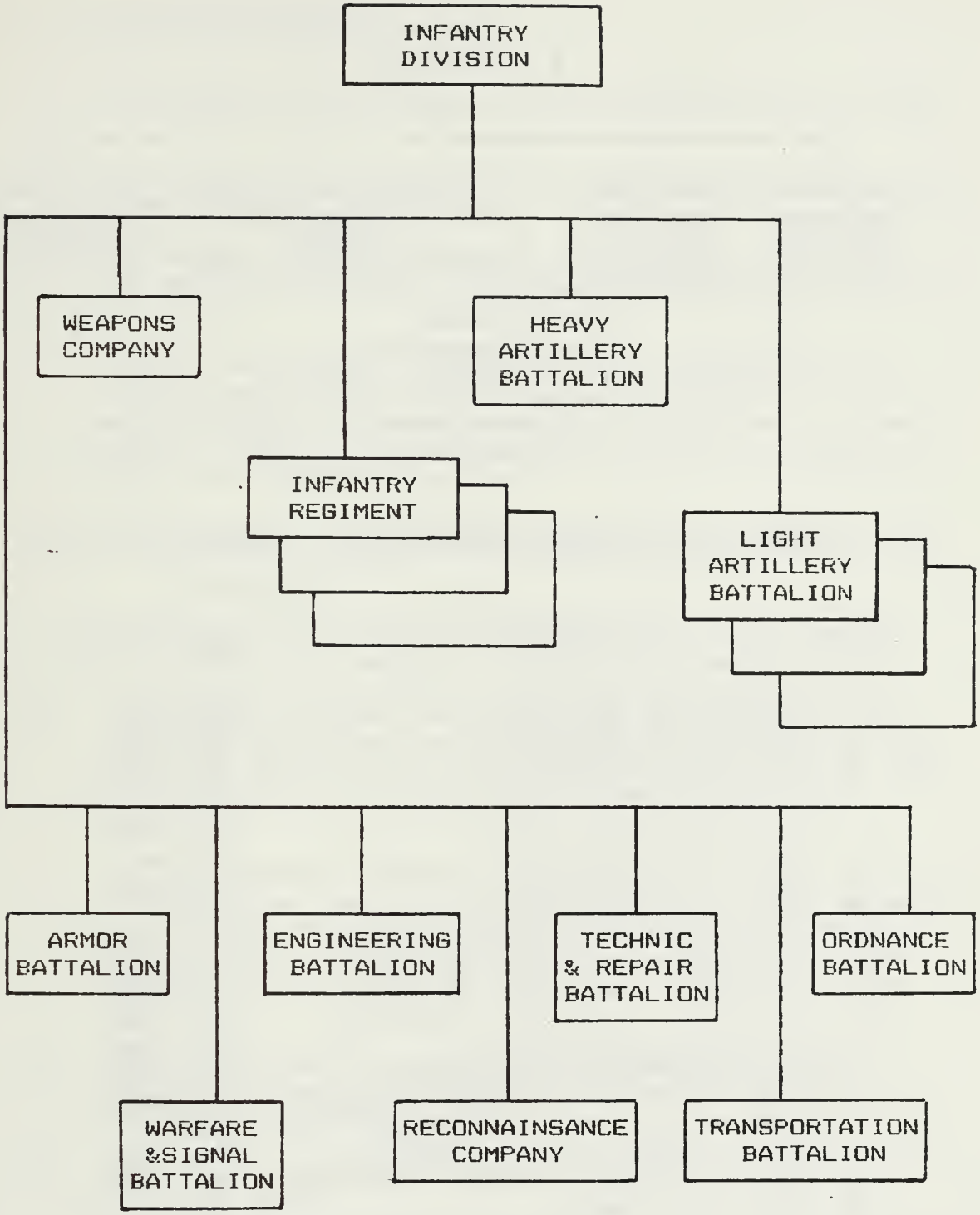


Figure 2.1: Organization of Infantry Division

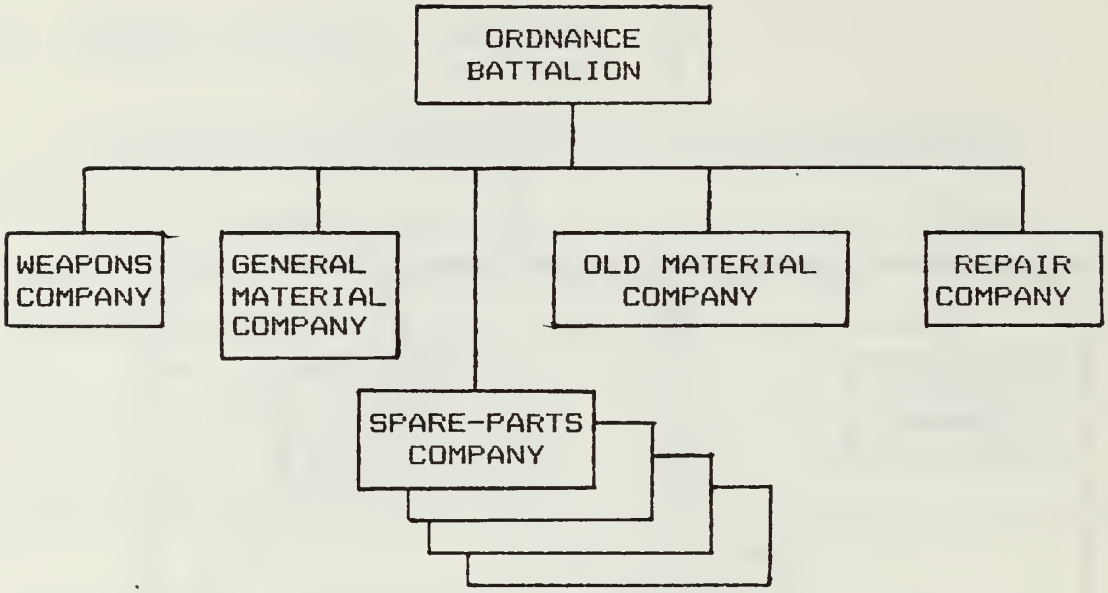


Figure 2.2: Organization of an Ordnance Battalion

Seq#	Unit	Abreviation
1	511 Rifle Battalion	511 RB
2	512 Rifle Battalion	512 RB
3	513 Rifle Battalion	513 RB
4	521 Rifle Battalion	521 RB
5	522 Rifle Battalion	522 RB
6	523 Rifle Battalion	523 RB
7	531 Rifle Battalion	531 RB
8	532 Rifle Battalion	532 RB
9	533 Rifle Battalion	533 RB
10	110 Light Artillery Battalion	110 LAB
11	111 Light Artillery Battalion	111 LAB
12	112 Light Artillery Battalion	112 LAB
13	150 Heavy Artillery Battalion	150 HAB
14	250 Armor Battalion	250 ARB
15	750 Engineering Battalion	750 ENB
16	450 Warefare & Signal Battalion	450 WSB
17	350 Reconnainsance Company	350 RC
18	650 Transportation Battalion	650 TRB
19	850 Technical & Repair Battalion	850 T&RB
20	Weapons Company	WC

Figure 2.3: List of units of an Infantry Division

3. Basic Operations

The basic operations of the Ordnance Battalion are:

a. The Ordnance Battalion has an organizational structure as depicted in Figure 2.2.

b. The GENERAL MATERIAL COMPANY of the Battalion holds all the general materials and supports all units of the Division.

c. The four SPARE PART COMPANIES of the Battalion are independent of each other and each contains approximately the same kinds and quantities of spare-parts. The division of responsibilities of the four SPARE PART COMPANIES are:

   (1) The first Company supplies all the units of the Division except those belonging to the Infantry Regiments.

   (2) Each of the other three Companies supplies all units of a specific Infantry Regiment.

d. The WEAPONS COMPANY does not have any relation with materials and contains all the military and civilian personnel needed for the different activities of the Ordnance Battalion other than those of handling materials.

e. The OLD MATERIAL COMPANY receives all the old materials returned from the units of the Division. Its purpose is to separate these materials into repairables and useless. The useless materials are destroyed, and the repairables are given to the Repair Company of the Battalion.

f. The REPAIR COMPANY receives all repairable old materials from the Old Material Company of the Battalion and repairs them.

g. The operational and tactical activities of the Battalion are:

   (1) The GENERAL MATERIAL COMPANY and the first SPARE PART COMPANY remain with the Ordnance Battalion supporting the Infantry Division.

(2) Each of the other three SPARE PART COMPANIES follows its supporting Regiment and acts independently from the Ordnance Battalion.

4. Documents

Two documents are used:

a. The first document is used by all units of the Infantry Division in order to request materials from the Ordnance Battalion.

b. The second document is used by the Ordnance Battalion in order to give the requested materials to the units.

5. Acquisition of Materials

Once a week the Ordnance Battalion informs the Ordnance Regiment with its required quantities of materials. In order to calculate the required quantity in one material the following factors must be taken into consideration:

a. The existing quantity of the particular material.

b. The maximum and minimum quantity.

c. The owed quantity.

d. The required time between the ordered date and the date of receiving the materials.

The required time between ordering a material and receiving it is about three months, so it is crucial to calculate correctly the ordered quantity.

B. SYSTEM OBJECTIVES

1. Reasons for the Project

The reasons for switching from the existing manual system to an automated one are:

a. To achieve greater processing speed.

Using the computer's inherent ability to calculate, sort, and retrieve data and information is faster than that of people doing the same tasks.

b. To achieve better accuracy and improve consistency.

Carrying out computing steps, including arithmetic, correctly and in the same way each time.

c. To achieve faster information retrieval.

Locating and retrieving information from storage.

d. To reduce the cost.

Using computing capability to process data at a lower cost than is possible with other methods, while maintaining accuracy and performance levels.

e. To provide better security.

Safeguarding sensitive and important data andprograms in a form that is accessible only to those persons having authorization. DBMS provides several security features such as : User authorization, Program authorization, Subschema authorization (i.e. area, records, sets).

f. To save personnel for other activities.

A manual system requires a large number of people each performing all activities manually. A computerize system, on the other hand, requires fewer personnel with knowledge in programming and the use of a computer.

2. Project Tasks

The project tasks are the following:

a. To keep track of:

   (1) The existing materials in the Ordnance Battalion.

   (2) Delivered materials to each unit of the Infantry Division.

   (3) Owed materials to each unit.

b. To provide lists of materials and units under several specifications. For example, lists of materials which entered the Battalion on a specific date or during a specific period of time, lists of materials given to a unit on a particular date or period of time or by a specific registration number, or lists of units supplied by a particular material, etc.

c. To make inventory reordering and ensure that adequate quantities of materials are on hand and available for use without carrying an excessive, and therefore costly, quantity.

d. To provide an automatic system of supplying the owed materials to the units as soon as these materials are made available to the Battalion.

e. To ensure insertions, deletions, and modifications in all the files of the system are made quickly and accurately.

f. To provide information such as:

   (1) Which materials are needed for the repair of an item of general material (for example, for the vehicle M35).

   (2) The location of each material in the store.

   (3) The materials that are under the safety limit on quantity, or up to the maximum amount, or under the minimum quantity limit, etc.

C. AUTOMATION ENVIRONMENT

Computing services for the Ordnance Battalion will be provided by a microcomputer, IBM PC or a corresponding compatible using a DBASE II package software. The microcomputer will be of 256 k memory and it will have two floppy disks, and one harddisk of 20 megabytes. Inventory control is the major application of the development effort.

1. User Requirements

A requirement is a feature that must be included in a new system and may include a way of capturing or processing data, producing information, controling a business activity, or supporting management. The determination of requirements thus means studying the existing systems and collecting details about them to determine these requirements.

In our system, the answers to the following two questions identify the basic requirements: What is the basic process? What data is used or produced during that process?

a. Basic Process of the System

Inventory reordering and modifications are the main processes of the project.

The tasks of the system have been described above in paragraph II.B

The required steps to perform reordering are: verifing the stock on hand, determining future requirements and ideal time to place orders, determining quantities to order, and placing the order.

All of the above steps take place in the logistic component of each company of the Ordnance Battalion, using information provided by the inventory, and owed files of the system, and the requested quantity of each material, used to determine the current levels, which then are compared to statistical elements based on the levels of personnel and main materials of the Infantry Division.

The estimating time for this process as well as for any other computing activities is a few minutes for simple and routine orders, and longer for orders involving new logistical support under special circumstances.

The process for reordering will take place once a week. Any other process will be performed when required.

The information produced is used by any authorized person of the Ordnance Battalion or Infantry Division.

b. Input and Output Information

(1) Input Information. The needed input information is organized as follows:

Each material must have a Part Number of the form (0000-000-0000), a material name and belongs in one of the categories, Transportation means, Communication means, Weapons, Other-General-Material, and spare-parts. Figures 2.4, 2.5, 2.6, 2.7, and 2.8 show materials of the above categories, respectively.

(2) Documents. Basically two working documents are used to request and supply materials.

(a) The first is the **REQUEST-DOCUMENT** as shown in Figure 2.9. This document is completed by the

supplied unit and submitted to the supplying battalion. Each document may contain up to 10 entries of General materials, or Spare parts. The Spare parts requested by each document must be used in only one general material.

(b) The second and most important document is the SUPPLIES-DOCUMENT, Figure 2.10. This document is completed by the supplying unit and is published in two copies, one for supplying unit and the other for the supplied unit. The SUPPLIES-DOCUMENT is the official promissary note for the unit that takes the materials and the official document of credit for the Ordnance Battalion.

This document is also used by the Ordnance Battalion to subtract from its store the quantity of the given materials, and by the unit to add in its store the materials received.

The following information is necessary:

(1) The number of personnel of each unit as well as the total personnel of the Infantry Division.

(2) The total number of each main material existing in each unit.

Also two books are used by the Ordnance Battalion:

(1) One for registration enter documents.

(2) The second for registration exit documents

(3) Output Information. The database system can be applied to any Ordnance Battation in any Infantry Division in the Greek Armed Forces. The following information is provided:

(a) List of materials received by the Battalion on a particular date.

(b) List of all materials received by the Battalion in a specific period of time.

(c) List of all materials received by the Battalion based on a particular registration number.

(d) List of material(s) given to a unit on a specific date or in a specific period of time.

(e) List of all materials needed reorder

(f) List of materials owed to a unit.

(g) List of all units to which a specific material is owed.

(h) List of all spare parts used for repair from a specific main material.

Part-no	Material name	Qtity	Max	Min	Ind	Owed	Wait
1001-000-0000	Vehicle M34	35	50	20	30	0	0
1002-000-0000	Vehicle M35	25	50	20	30	0	25
1003-000-0000	Vehicle M38	5	20	5	10	10	25
1004-000-0000	Vehicle M38A1	5	15	5	10	5	20
1005-000-0000	Vehicle M41	0	5	2	3	3	8
1006-000-0000	Vehicle M51	2	5	1	3	0	2
1007-000-0000	Vehicle M113	1	3	1	2	1	2
1008-000-0000	Vehicle AMX-10	1	3	1	2	0	1
1009-000-0000	Vehicle Leopard	1	3	1	2	2	3
1010-000-0000	Vehicle M6000	5	20	5	10	5	25
1011-000-0000	Vehicle D/G 3/4	0	10	3	5	15	25
2001-000-0000	Armor M47	1	2	1	1	0	1
2002-000-0000	Armor M48	0	2	1	1	1	3
2003-000-0000	Armor AMX-30	1	2	1	1	1	2
2004-000-0000	Armor M60	0	2	1	1	0	1
2005-000-0000	Armor M47A1	0	2	1	1	2	3
2006-000-0000	Armor M48A1	1	2	1	1	0	1
2100-000-0000	Armor Veh M453	1	2	1	1	1	2

Figure 2.4: Typical List of Transportation Means

Communication Means

Part-no	Material name	Qtity	Max	Min	Ind	Owed	Wait
3001-000-0000	Wireless PRC-9	5	10	5	7	1	5
3002-000-0000	Wireless SCR-536	2	20	10	15	5	23
3003-000-0000	Wireless PRC-10	5	10	5	7	0	3
3004-000-0000	Wireless PRC-11	0	5	2	3	2	5
3005-000-0000	Wireless VRC-30	2	5	2	3	0	2
3006-000-0000	Wireless VRC-100	2	5	2	3	0	2
3007-000-0000	Wireless VRC-150	2	5	2	3	1	2
3008-000-0000	Telephone A-10	5	15	5	10	5	15
3009-000-0000	Telephonet B-15	0	5	2	3	1	4
3010-000-0000	Telephonet C-20	5	15	5	10	0	10
3011-000-0000	TV set T-31	0	2	1	1	2	3
3012-000-0000	TV set T-55	1	2	1	1	1	3
3020-000-0000	Transmiter TS-50	2	2	1	1	0	0
3021-000-0000	Receiver RS-100	2	5	2	3	1	4
3022-000-0000	Receiver RS-150	1	3	1	2	0	2
3023-000-0000	Transmiter TS-55	1	3	1	2	0	2

Figure 2.5: Typical List of Communication Means

Part-no	Material name	Qtity	Max	Min	Ind	Owed	Wait
4001-000-0000	Rifle M1 .30	50	100	50	70	0	20
4002-000-0000	Rifle M1A1 .30	10	20	10	15	5	15
4003-000-0000	Pistol .38	5	10	5	7	0	5
4004-000-0000	Pistol .45	10	10	5	7	0	0
4005-000-0000	Machine-Gun .45	1	5	2	3	1	3
4006-000-0000	Machine-Gun .30	1	5	2	3	1	3
4007-000-0000	Machine-Gun .50	1	5	2	3	1	3
4008-000-0000	Gun 105 mm	2	5	2	3	0	3
4009-000-0000	Gun 155 mm	3	5	2	3	0	0
4010-000-0000	Gun 100 mm	2	5	2	3	0	2
4011-000-0000	Gun 8'	1	5	2	3	1	5
4012-000-0000	Gun 90 mm	1	5	2	3	0	4
4013-000-0000	Gun 205 mm	1	5	2	3	0	4
4014-000-0000	Rifle .303	20	50	10	20	0	0
4015-000-0000	Pistol .22	2	5	2	3	0	2
4016-000-0000	Rifle .22	15	50	20	30	5	35

Figure 2.6: Typical List of Weapons

Other-General-Material

Part-no	Material name	Qtity	Max	Min	Ind	Owed	Wait
9000-101-0001	Beds	25	50	20	30	0	20
9000-101-0002	Blackets	200	500	50	200	0	0
9000-102-0001	Shirts	150	200	50	100	0	0
9000-102-0002	Socks	50	500	50	100	0	400
9000-102-0003	Shoes	50	100	30	50	0	50
9000-102-0004	Pantalon	10	100	30	50	20	100
9000-103-0001	Dishes	10	200	50	100	0	200
9000-103-0002	Pots	10	200	50	100	10	200
9000-104-0001	Tables	15	50	10	20	0	0
9000-104-0002	Chairs	15	50	10	20	0	30
9000-199-0001	Soaps	20	100	20	30	10	80
9000-199-0002	Toothbranshes	30	100	30	50	0	50
9000-199-0003	Hats	20	200	50	100	20	200
9001-001-0001	Knife	20	100	20	50	0	50
9001-001-0002	Fork	15	100	20	50	10	70
9001-001-0003	Spoon	15	100	20	50	10	85
9100-100-0001	Oil filtr Wrench	5	10	5	7	0	3
9100-100-0002	Socket Wrench st	3	10	5	7	2	5
9100-100-0003	Extention Bar	2	5	2	3	0	3
9100-100-0004	Socket	3	5	2	3	0	0
9100-100-0005	Spark Plg Socket	5	5	2	3	0	0
9200-100-0001	Hammer Holder	4	10	3	5	0	0
9200-100-0002	Saw	3	10	3	5	0	0
9200-100-0003	Nail set	4	10	3	5	0	0
9200-100-0004	Hacksaw	2	10	3	5	2	0
9200-100-0005	Window Scraper	2	10	3	5	0	0
9200-100-0006	Plumb Bob	1	10	3	5	2	0
9200-100-0007	Clamp	0	10	3	5	4	0
9300-100-0001	Wrench 1'	2	10	3	5	0	5
9300-100-0002	Wrench 2'	3	10	3	5	3	0
9300-100-0003	Wrench 5'	4	10	3	5	0	3
9300-100-0004	Wrench 10'	2	10	3	5	0	3
9300-100-1001	Screw Driver 1'	1	10	3	5	2	0
9300-100-1002	Screw Driver 1,5	5	10	3	5	0	0
9300-100-1003	Screw Driver 2'	10	10	3	5	0	0
9300-100-1004	Screw Driver 3'	8	10	3	5	0	0
9300-100-1005	Screw Driver 4'	7	10	3	5	0	0
9300-100-1006	Screw Driver 5'	3	10	3	5	4	0
9300-100-1007	Screw Driver 10'	4	10	3	5	0	3

Figure 2.7: Typical List of Other-General -Material

S P A R E P A R T S

Part-no	Material name	Qtity	Max	Min	Ind	Owed	Wait
1002-100-0001	Starter	1	5	2	3	3	2
1002-100-0002	Spark Plug	10	50	20	30	10	0
1002-100-0003	Ignition Wire St	3	5	2	3	0	0
1002-100-0004	Battery Repr Kit	5	5	2	3	0	0
1002-100-0005	Battery Tester	2	5	2	3	1	3
1002-100-0006	Battery Pole	3	5	2	3	0	2
1002-100-0007	Battery Cable	4	5	2	3	0	0
1002-100-0008	Battery Terminal	1	5	2	3	3	5
1002-200-0001	Automobil V-Belt	3	5	2	3	0	2
1002-200-0002	Washer Pump	2	5	2	3	1	0
1002-300-0001	Fluid Monitor	1	5	2	3	3	5
1002-300-0003	Wiper Delay	4	5	2	3	0	0
1002-400-0001	Radiator Cap	3	5	2	3	0	0
1002-100-0009	Thermostat	2	5	2	3	1	3
1002-400-0002	Fuel line Hose	3	5	2	3	0	0
1002-400-0003	Heater hose	2	5	2	3	1	0
1002-900-0001	Compressed Asbes	1	5	2	3	3	2
1002-500-0001	Multiplex Weld	2	5	2	3	1	5
1002-400-0004	Oil Filter	3	5	2	3	0	2
1002-500-0002	Air Filter	2	5	2	3	1	3
1002-600-0001	Booster Cable	4	5	2	3	1	3
1002-600-0002	Hold Trap	1	5	2	3	2	0
1001-100-0001	Starter	23	5	2	3	1	2
1001-100-0002	Spark Plug	45	50	10	20	0	20
1001-100-0003	Ignition Wire St	2	5	2	3	0	2
1001-100-0004	Battery Cable	1	5	2	3	2	2
1001-200-0001	Automobil V-Belt	4	5	2	3	0	2
1001-300-0001	Radiator Cap	1	5	2	3	0	2
1001-100-0005	Thermostat	2	5	2	3	0	2
1001-400-0001	Oil Filter	1	5	2	3	0	2
1001-500-0001	Air Filter	0	5	2	3	0	2
1003-100-0001	Starter	2	5	2	3	0	2
1003-100-0002	Spark Plug	5	5	2	3	0	2
1003-100-0003	Ignition Wire St	3	5	2	3	1	0
1003-100-0004	Battery Cable	1	5	2	3	0	2
1003-200-0001	Automobil V-Belt	1	5	2	3	1	3
1003-300-0001	Radiator Cap	4	5	2	3	0	4
1003-100-0005	Thermostat	1	5	2	3	2	0
1003-400-0001	Oil Filter	1	5	2	3	0	2
1003-500-0001	Air Filter	3	5	2	3	0	2

Continue next page

Part-no	material name	Qtity	Max	Min	Ind	Owed	Wait
1004-100-0001	Starter	2	5	2	3	0	2
1005-100-0001	Starter	4	5	2	3	0	2
1006-100-0001	Starter	2	5	2	3	0	2
1007-100-0001	Starter	0	5	2	3	0	2
1010-100-0001	Starter	1	5	2	3	0	0
1004-100-0002	Spark Plug	6	50	20	30	0	20
1004-100-0003	Ignition Wire St	2	5	2	3	0	0
1004-100-0004	Battery Cable	1	5	2	3	1	0
1004-200-0001	Automobil V-Belt	2	5	2	3	0	0
1004-400-0001	Radiator Cap	1	5	2	3	0	2
1004-100-0005	Thermostat	3	5	2	3	1	3
1004-400-0001	Oil Filter	2	5	2	3	0	3
1004-500-0001	Air Filter	2	5	2	3	0	0
3001-100-0001	Lamp L-10	2	5	2	3	0	1
3002-100-0001	Lamp L-1	2	10	5	7	0	2
3003-100-0001	Lamp L-2	1	10	3	5	0	1
3005-100-0001	Lamp L-25	3	5	2	3	0	0
3006-100-0001	Lamp L-35	2	5	2	3	0	0
3001-100-0002	Transistor	2	10	3	5	0	1
3002-100-0002	Transistor	1	5	2	3	2	0
3003-100-0002	Transistor	5	20	5	10	5	5
3004-100-0002	Transistor	10	50	10	5	0	0
3005-100-0002	Transistor	5	10	3	5	0	0
9300-200-1001	Plier 8'	2	10	3	5	0	5
9300-200-1002	Plier 9'	1	10	3	5	2	0
9300-200-1003	Plier 10'	5	10	3	5	0	0
9300-200-1004	Plier 20'	7	10	3	5	0	0
9400-000-0001	Hydraul Jack 1 t	1	10	3	5	1	10
9400-000-0002	Hydraul Jack 2 t	4	10	3	5	0	4
9400-000-0003	Hydraul Jack 5 t	7	10	3	5	0	0
9400-000-0004	Hydraul Jack 10t	2	10	3	5	1	7
9400-000-0005	Hydraul Jack 30t	1	10	3	5	0	5
4001-000-0001	Brooch	10	10	3	5	0	0
4002-000-0001	Brooch	2	5	2	3	0	0
4003-000-0001	Brooch	1	3	1	2	0	0
4004-000-0001	Brooch	2	3	1	2	0	0
4005-000-0001	Brooch	1	2	1	1	0	0
4006-000-0001	Brooch	0	2	1	1	1	0
4001-000-0002	Butt-end	3	3	1	2	0	0
4002-000-0002	Butt-end	1	5	2	3	1	0
4003-000-0002	Butt-end	0	3	2	1	0	1
4004-000-0002	Butt-end	0	3	2	1	0	0

Figure 2.8: Typical List of Spare-Parts.

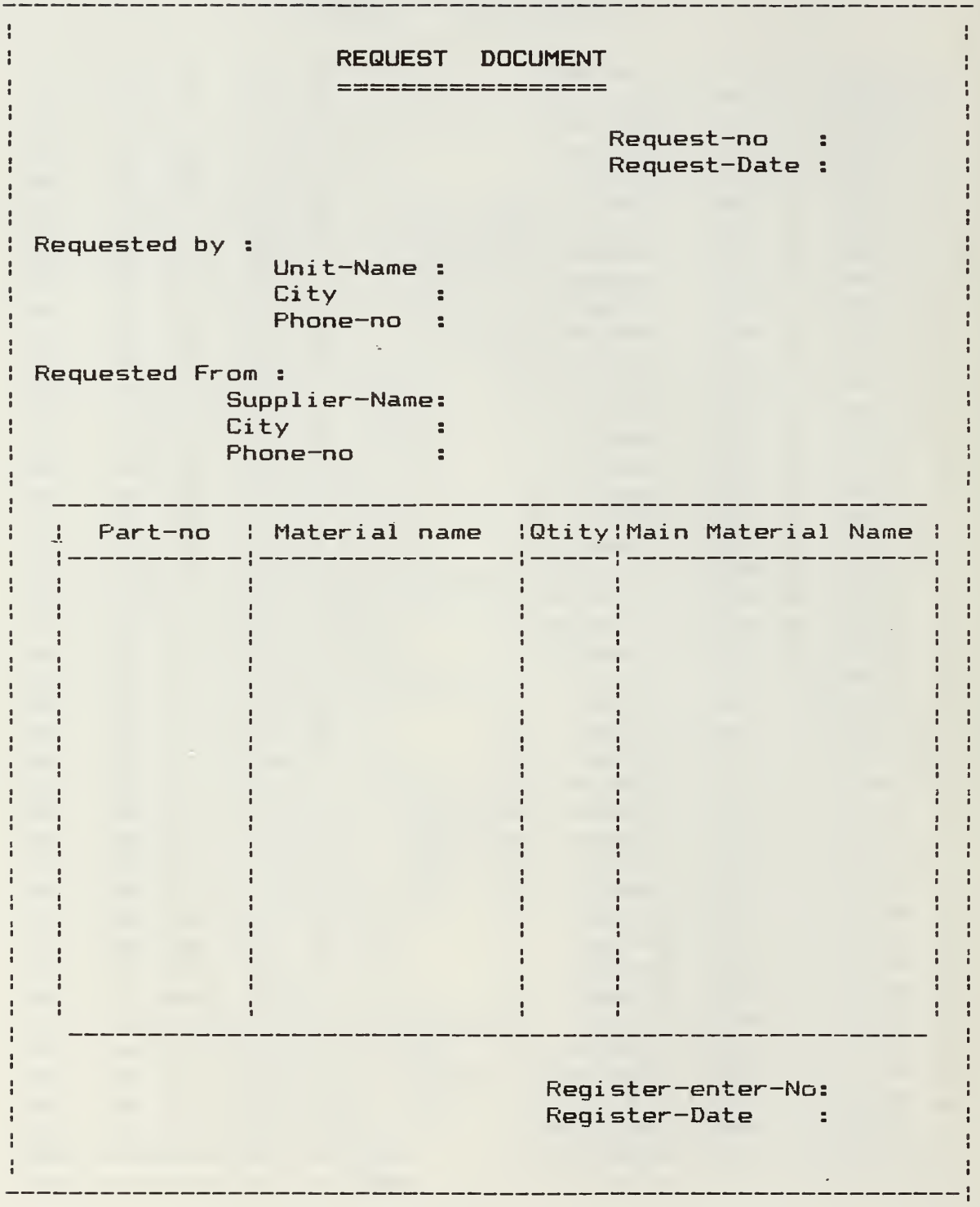


Figure 2.9: Request Material Document

SUPPLIER DOCUMENT

Reg exit no :
Reg exit Date:

Supplied by :
Supplier-name:
City :
Phone-no :

Supplied to :
Unit-name :
City :
Phone-no :

Seq#	Part-no	Material name	Qtity	Owed	Due-date

Reg-enter-no :

Reg-enter-Date:

Figure 2.10: Supplies Material Document

III. DESIGN PHASE

Generally speaking systems analysis and design refers to the process of examining a situation with the intent of improving it through better procedures and methods. This section overviews the system design while the system analysis has been presented in the previous section II.

Systems design is the process of planning a new system to replace or complement the old. But before this can be done, we must thoroughly understand the old system and determine how the computer can best be used to make its operation more effective. Once a decision is made, a plan is developed to implement the decision. The plan includes all systems design features, such as, new data capture needs, file specifications, operating procedures, and equipment and personnel needs. The systems design specifies all the features that are to be in the finished product. While analysis specifies **what** the system should do, design states **how** to accomplish the objectives.

Systems design proceeds through two phases: First the **logical design** and second the **physical design.**

This section summarizes the systems design activities started by identifying the logical design, that is, outputs, inputs, files, and functions of the system. Then the physical design is stated, that is, program software and working system.

A. LOGICAL DESIGN.

Logical design is concerned with the production of specifications for the new system, that is, the features of

the system, the outputs, the inputs, the files, and the functions of the system, all in a manner that meets the project requirements.

In our system, the specifications include reports and output-screen definitions describing stock on hand, and materials owed or given to several units of the Division. The logical design also specifies input forms and screen layouts of all transactions, and files for maintaining, stock data, transaction details, and supplier data. Function specifications describe methods to enter data, run reports, copy files, and detect programs, should they occur.

1. Design of System Output

Often the most important feature of an information system for users is the output it produces. Without quality output, they may even feel the entire system is so unnecessary that they avoid using it and thus possibly cause the system to fail.

The term 'output' is applied to any information produced by the system, whether printed, or displayed.

a. Output Methods

The system will use two methods for output, print output and display output.

(1) Print output. The printed output options will be the printed Reports and the special forms. The printed reports will be on standard size paper of 9 1/2 by 11 inches and the produced reports are shown on figure 3.1 . The system will also use a preprinted form as Figure 3.2, that is designed to include several special symbols, the trademark of the Battalion and it will be a multiple copies

Seq#	Description of printed Output Reports
1.	Material-report, which list all items received by the Battalion on a particular date, or a specific period of time, or with a particular registration number. This report includes the part-number, the material name and the quatity.
2.	Material-report, which list all the items given or owed to several units of the Division under particular considerations such as data, time, registration number etc. The reports give a description of the item (partno, matername, and the quantity)
3.	Unit-report, which list all units of the Division to which a particular material was given or supplied by the Battalion within a particular date.
4.	Information-report such as: A list of all spare parts needed to repair a main material (vehicle, weapon). A list of all material which are placed in a particular store or place. A list of all materials the quantity of which is under or up to a specific amount.
5.	Reorder-report, which list all material needed reorder. This report contains the part number, the material and the order quantity.

Figure 3.1: Table of Printed Output Reports.

document with a different color for each copy. This special form is of interleaved carbon copies.

(2) Display output. The use of displays is more convenient than the printer report because of the dropping cost of display equipment, the increase in the number of on-line systems, and the convenience display output offers to users.

SUPPLIER DOCUMENT

Regexitno :
Regexitdata:

Supplied by :
Supplier unit:
City :
Phone number :

Supplied to :
Unit name :
City :
Phone number :

Seq#	Partno	material name	Qtity	Owed	Due-date
1.					
2.					
3.					
4.					
5.					
6.					
7.					
8.					
9					
10.					

Regenterno :
Regenterdate:

Figure 3.2 : Preprinted Form for Computer Report.

In the Battalion, a display medium consisting of a display screen and a keyboard, will be used. The output will be displayed on a monitor, a high-resolution screen that can show details with extra clarity. All output data appearing on the screen of the display will be produced by the computer itself.

The size of the letters will be fixed, as is the capacity of the screen. The screen will have 80 columns of characters and 25 rows. The design of a display format enables the personnel of the logistic department of the Battalion to request and retrieve information about a specific material, by using inquiries.

b. Output Layout

Using the proper output medium and being sure that the output is calculated correctly does not guarantee that the reports, or displays will be useful. The layout does this.

An output layout is the arrangement of items on the printed output or the visual display. All output produced by the system will have a report title and headings on the layout sheet. The elements of each material (partno, matarial name, quantity etc) will be in one line and the same elements of different materials will be on the same column. Figure 3.3 shows some types of layout.

2. Design of System Input

All input of data and inquiries will be made using the computer display. Therefore, the development of the input display screens will result in input screens for instructing the system whether to produce output, accept data input, or permit editing of stored records. Display screens are also designed to input or edit data about individual items. In each case, information will be included on the display screens to instruct the user how to use the system and enter or receive the necessary information.

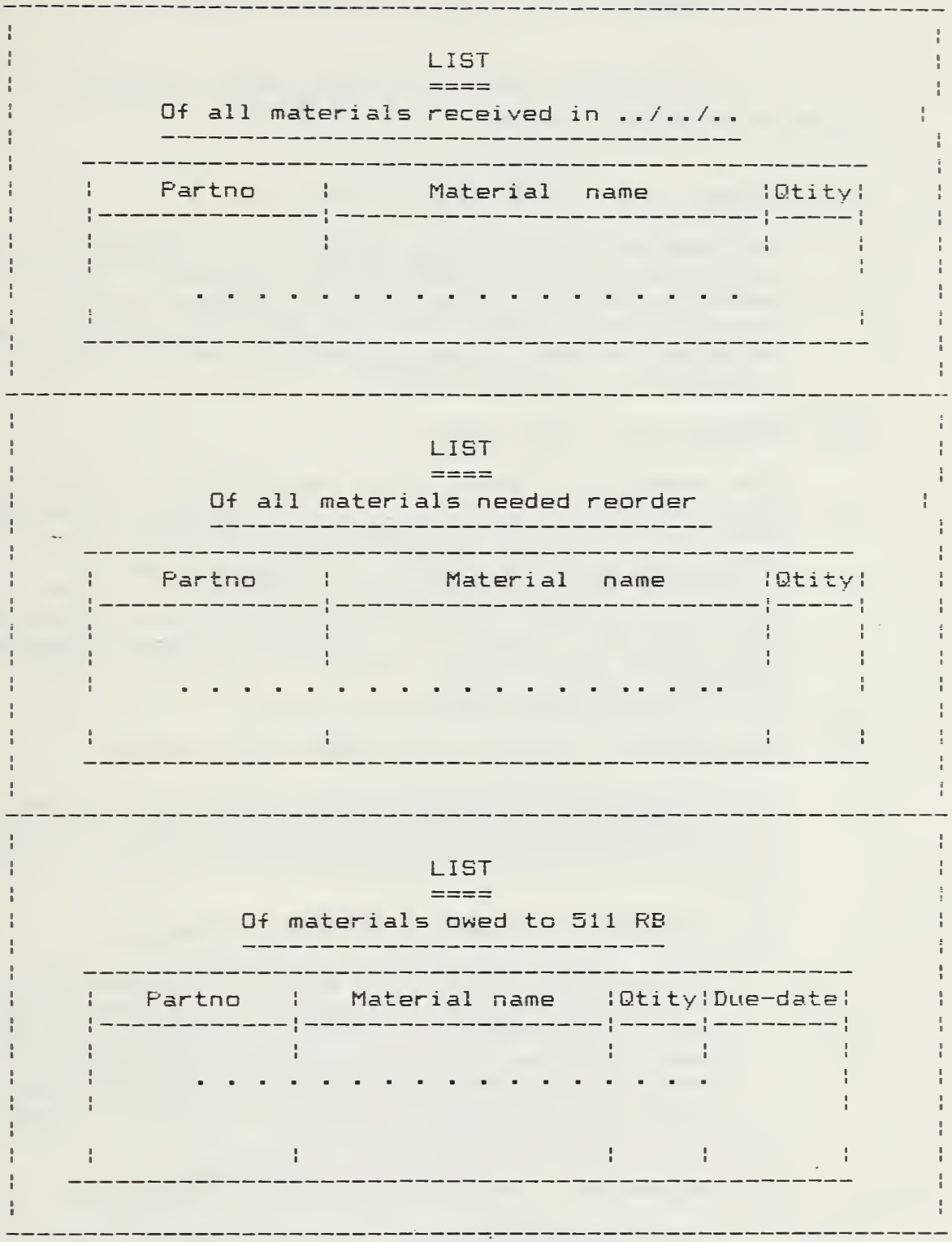


Figure 3.3: Output Layout Reports.

a. Design of our On-line Environment

The input design for the system includes the following display screens:

(1) Screen #1 : Enter the password.

The system asks the user to type the password in order to ensure that he is an authorized person. If the typed password is incorrect the process quits, otherwise the system asks the user to enter his name in order to check if he is authorized to retrieve the database. (Figure 3.4)

(2) Screen #2 : Main menu.

The screen includes a list, containing in one column the task-code from 0 to 4, and in another column the description of the available operations to the user. The user by typing codes 1 through 4 makes his selection of submenu1 through submenu4. Typing <0> exits the system. Finally typing a number other than the above or a symbol, an appropriate error message appears on the screen instructing the user how to continue. (Figure 3.5)

(3) Screen #3 : Update file functions (Submenu1)

In this screen the operations of insertion, deletion, and modification are described. The user by typing 1, 2, or 3 runs the corresponding sub-programs. (Figure 3.6).

(a) Screen #4 : Insert a material's quantity.

This program adds a new QUANTITY of materials in the INVENTORY file and modifies the fields WAITED, OWED, REGENTERNO, REGENTDATE, DUEDATE also of the same file, or creates a new record if the received material does not exits in the INVENTORY file. It also creates a new record of materials in the STATUS file. (Figure 3.7)

(b) Screen #5 : Delete a material's quantity.

This program subtract a given material from the field QUANTITY of INVENTORY file and modifies the

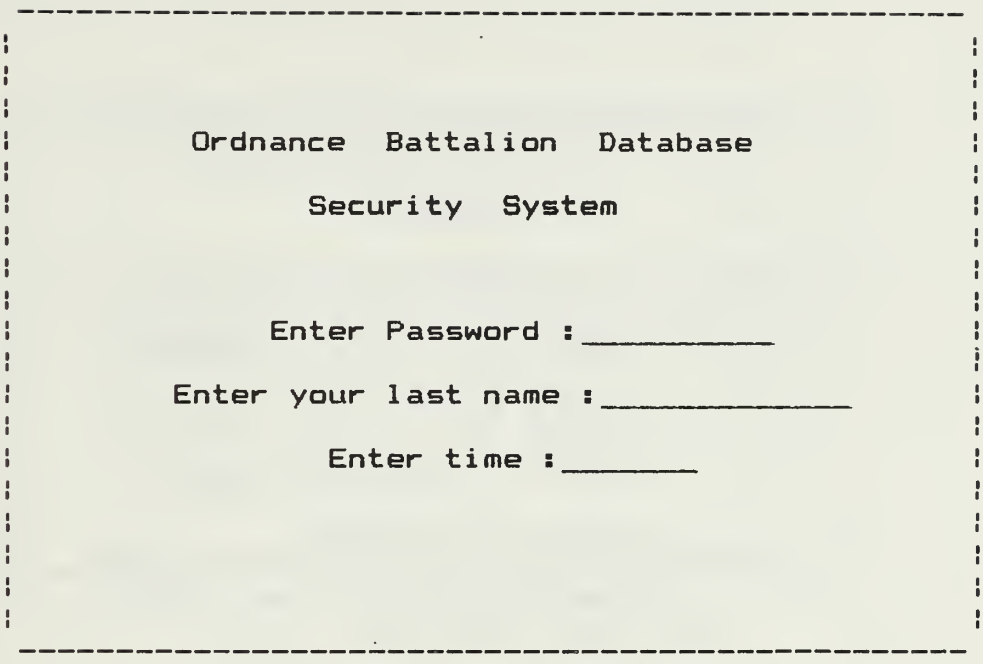


Figure 3.4 : Screen for Password.

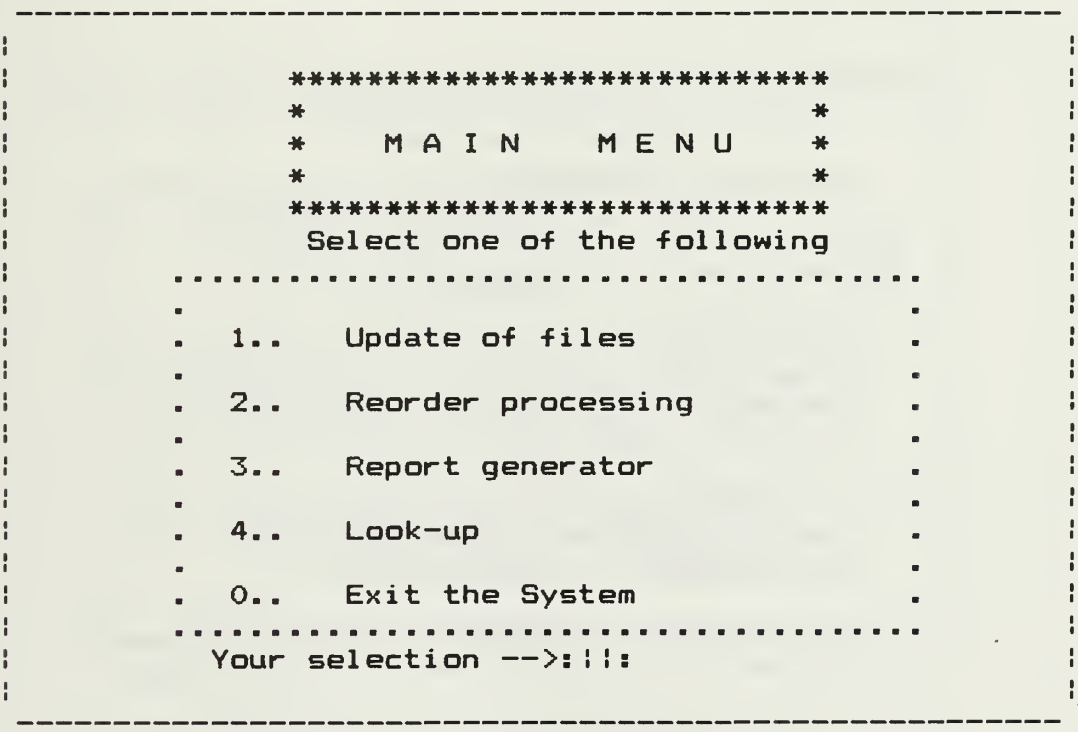


Figure 3.5 : Detail Screen for MAINMENU.

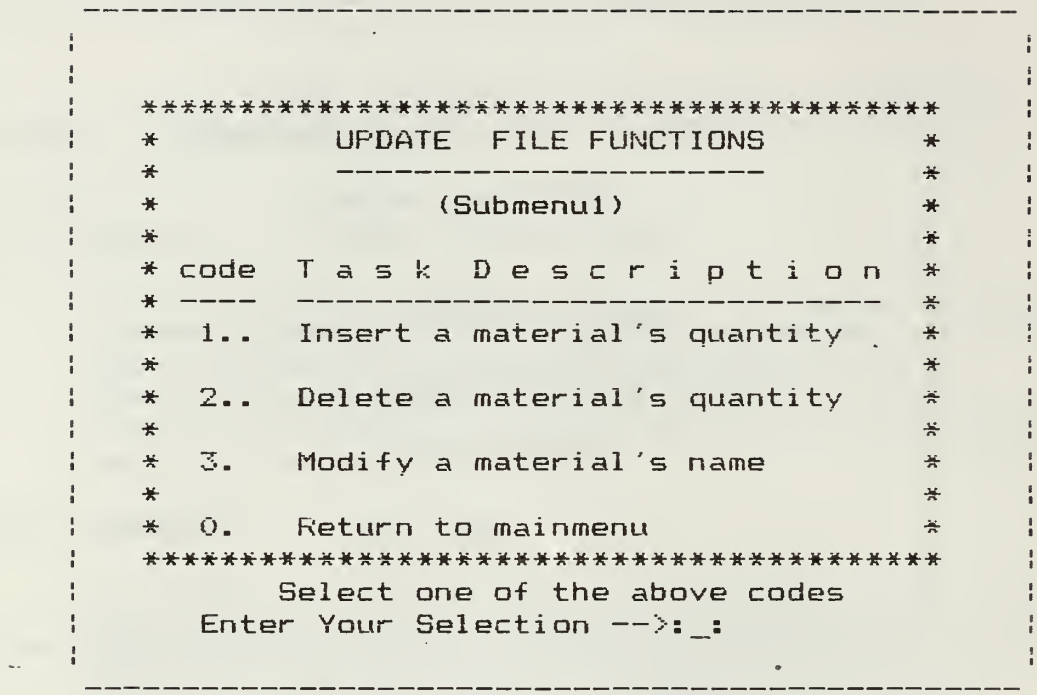


Figure 3.6 : Screen for SUBMENU1.

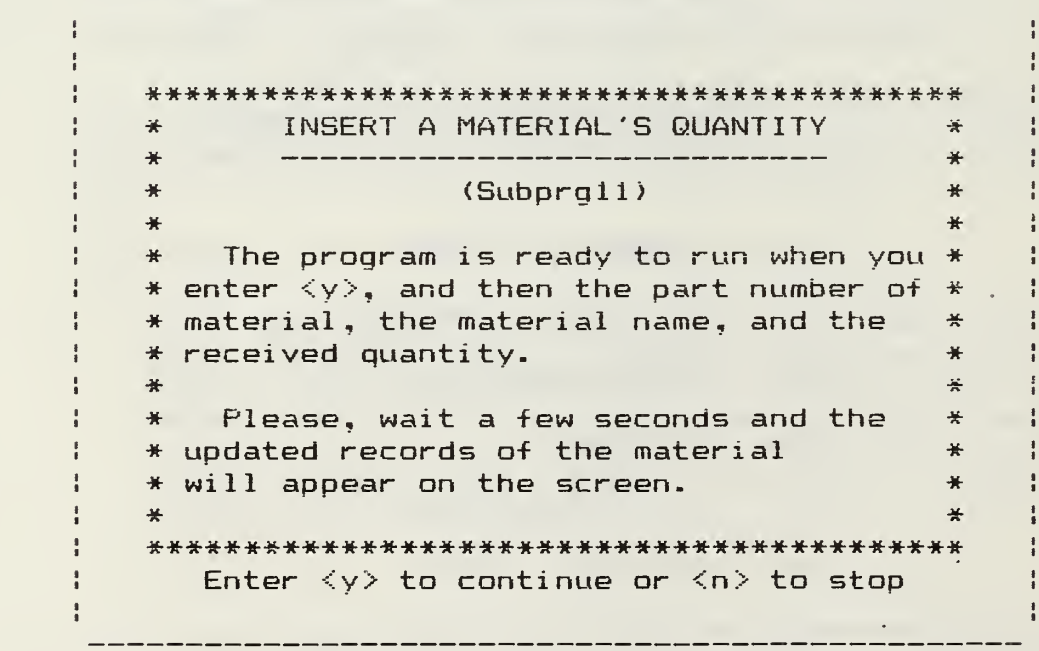


Figure 3.7 : Screen for SUBPRG11

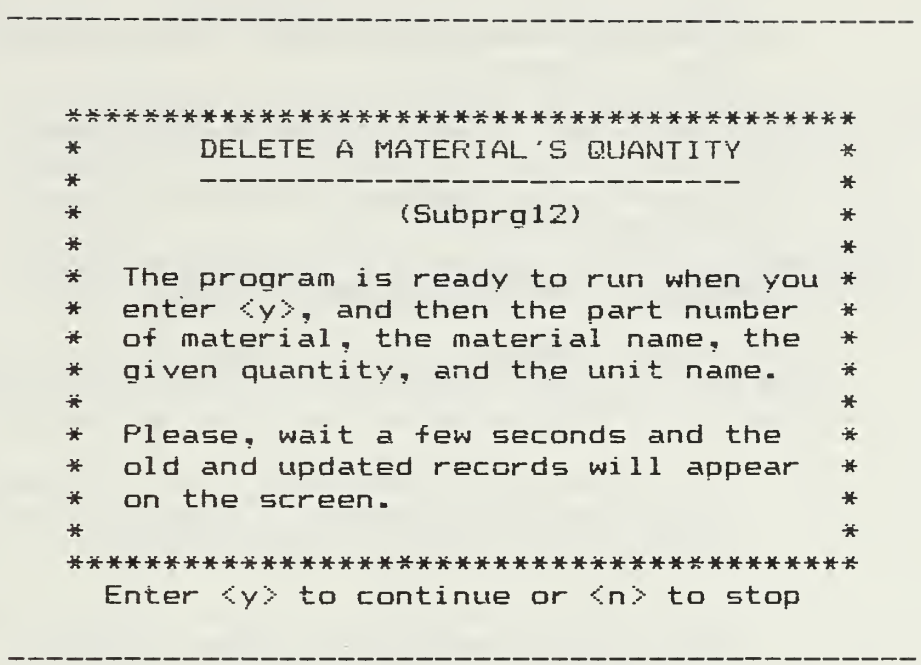


Figure 3.8: Screen for SUBPRG12

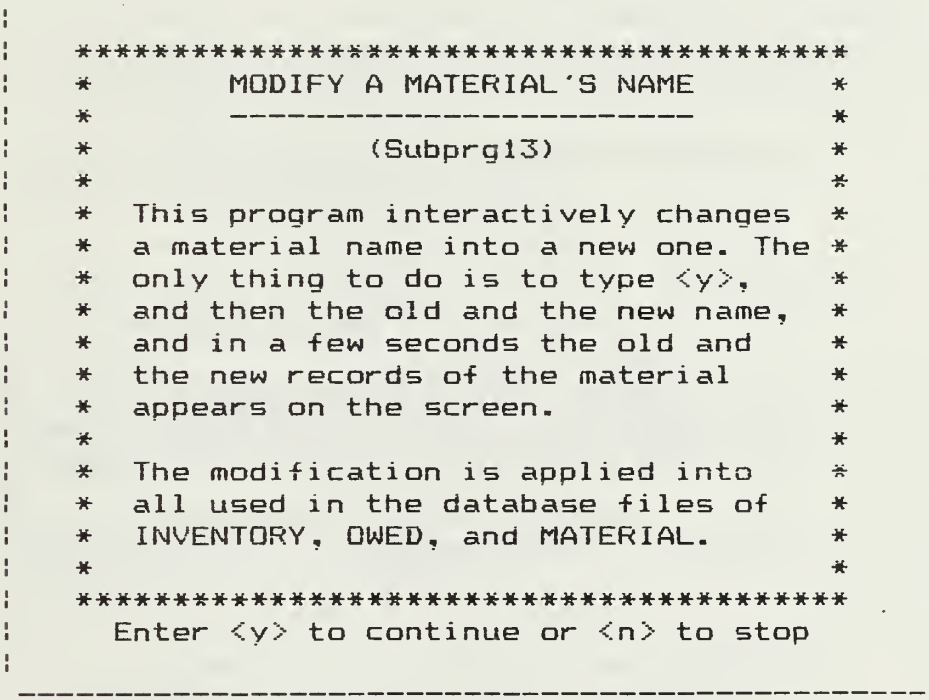


Figure 3.9: Screen for SUBPRG13

other fields if it is necessary. It also creates a new record of that material in the MATERIAL file. (Figure 3.8)

(c) **Screen #6 : Modify a material's name.**

This program changes a material name to a new one. (Figure 3.9)

(4) **Screen #7 : Reorder Reporting (Submenu2).**

This screen simple asks the user to type <y> if he wants to run the program or <n> to exit from submenu2. Especially this program does the following:

(a) It creates from INVENTORY file a temporary file (tmp1) which includes all materials the quantity of which is under the INDEX-ORDER.

(b) It creates from tmp1 file the ORDER file which contains all materials needed reorder.

(c) Finally it creates from ORDER file the tmp2 file for future purposes while at the same time it deletes the ORDER file. (Figure 3.10)

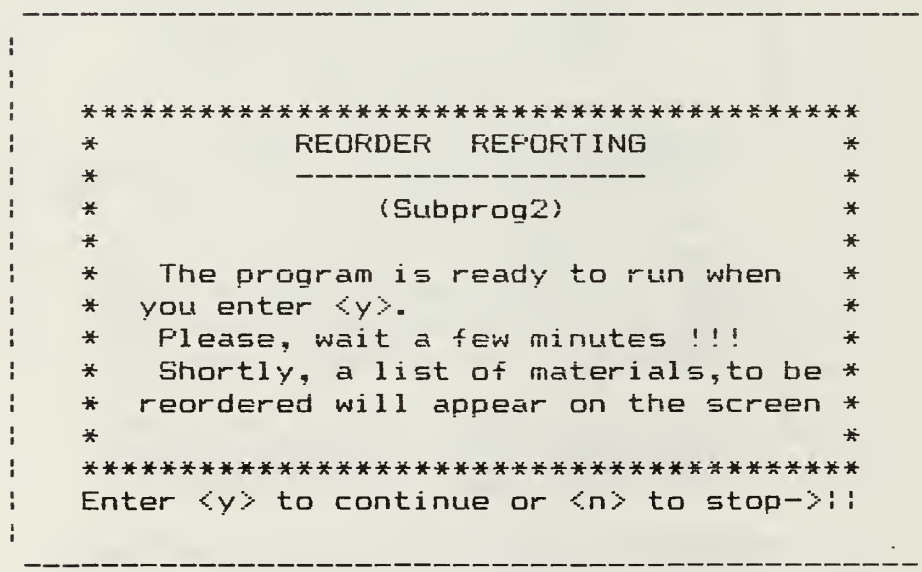


Figure 3.10: Screen for SUBPROG2

(5) Screen #8 : Report Gererator (Submenu3).

The screen includes a list of operations with the corresponding task-codes. The user has to type numbers 1 to 9 to select one of the described operations on the screen or <0> to return to mainmenu. The user typing a wrong number or symbol gets a message on the screen which gives him instructions on what to do. (Figure 3.11).

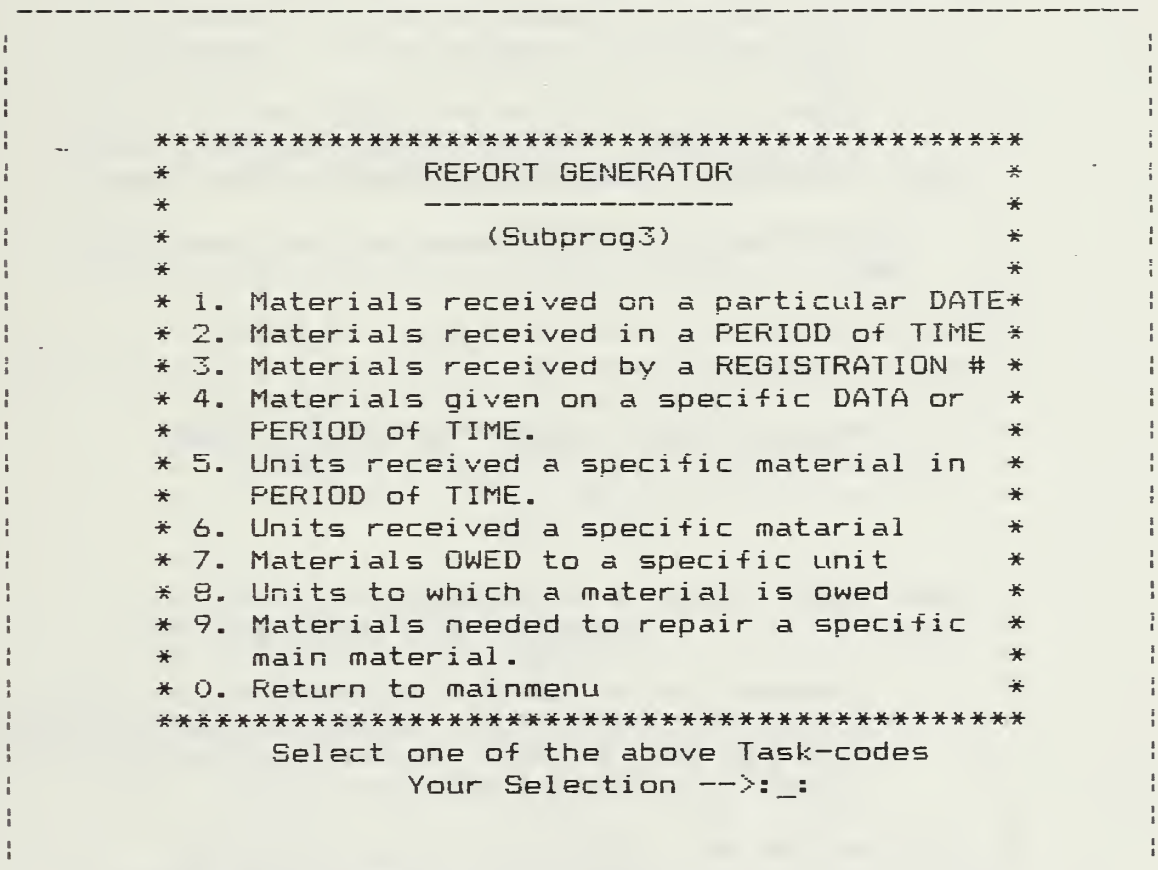


Figure 3.11 : Screen for SUBMENU3.

(a) Screen #9: Materials received on a DATE.

The user enters the specific date and on the screen appears a list of materials received by the Battalion on that DATE. (Figure 3.12)

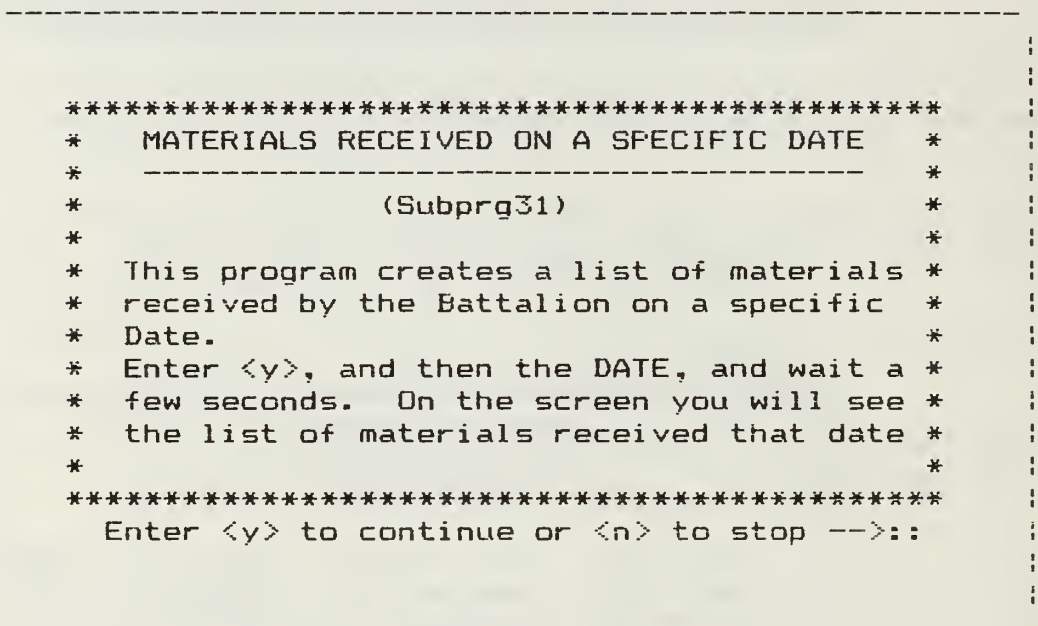


Figure 3.12 : Screen for Subprogram31

(b) Screen #10 : Materials received in a particular PERIOD of TIME.

The user has to enter the specific period of time and on the screen will appear a list of materials received by the Battalion in that period. (Figure 3.13)

(c) Screen #11: Materials received by a specific registration number

The screen gives intructions to the user on what to do. (Figure 3.14)

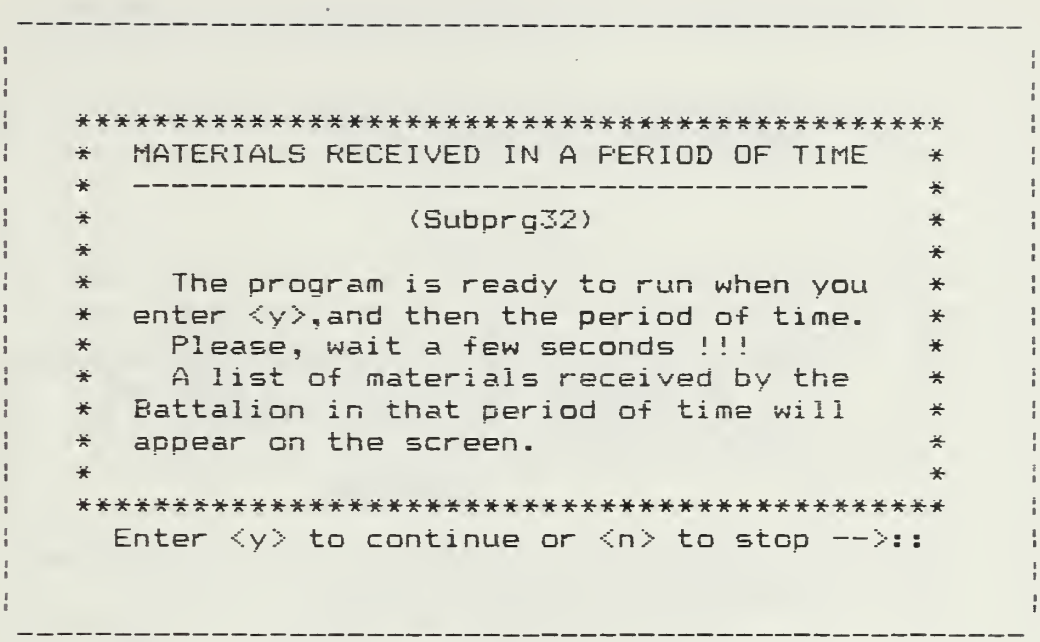


Figure 3.13: Screen for Subprogram32

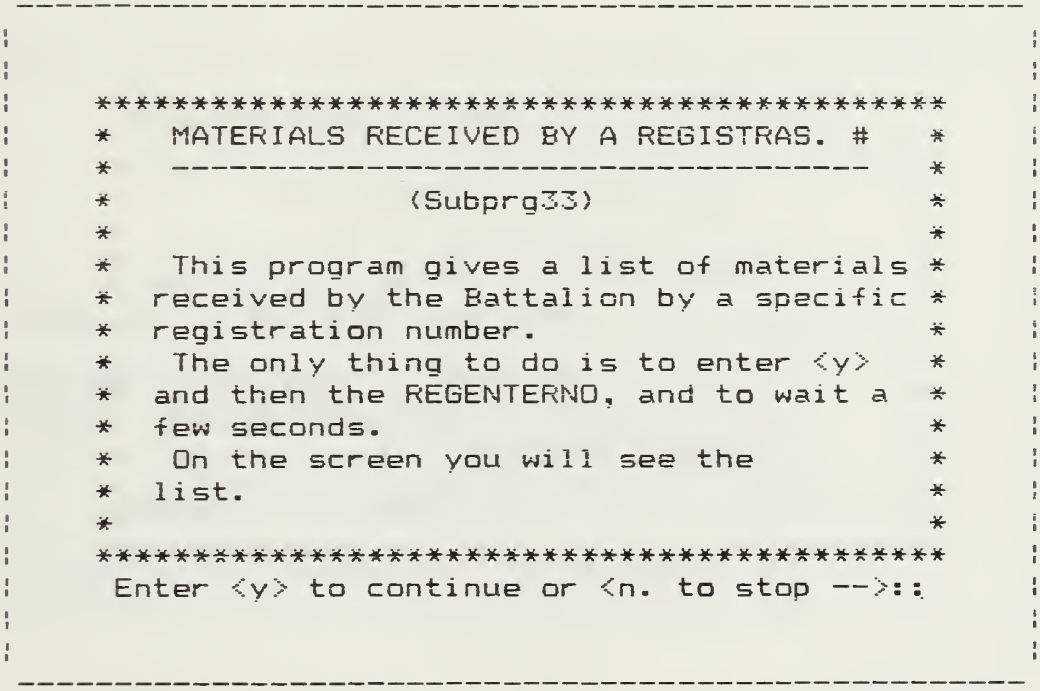


Figure 3.14: Screen for Subprogram33

(d) Screen #12: List of materials given on a specific DATE or in a PERIOD of TIME.

The user has to follow the instructions given on the screen. (Figure 3.15)

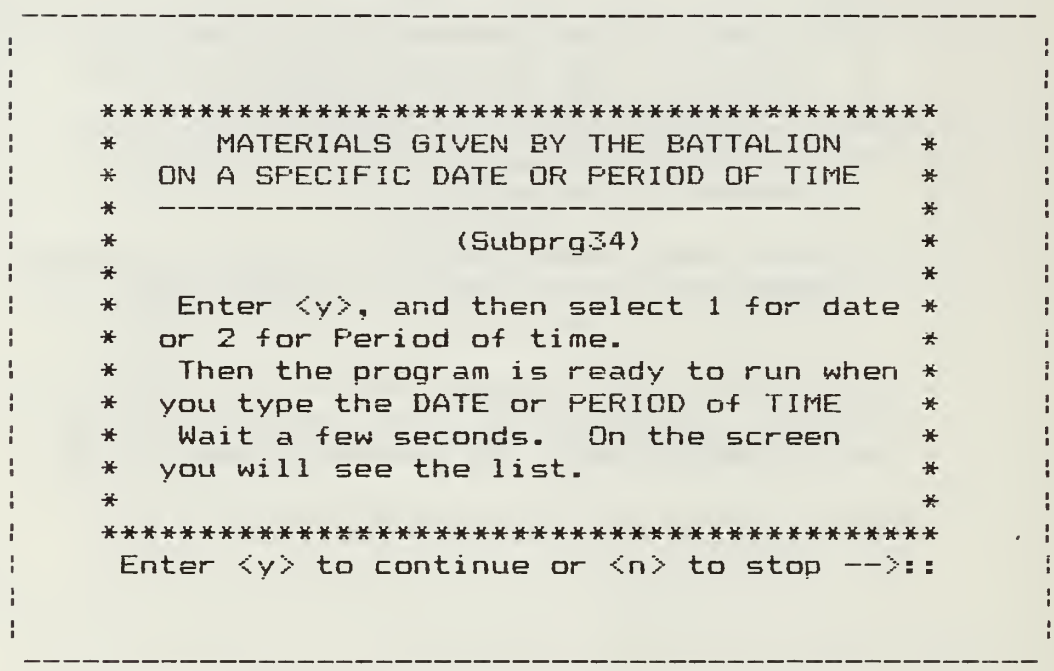


Figure 3.15: Screen for Subprogram34

(e) Screen #13: List of units received a material in a period of time

On this screen there are instructions to the user on what to do in order to make the desired list of units. (Figure 3.16)

(f) Screen #14: Units received a material.

The screen gives directions to the user on what to do. (Figure 3.17)

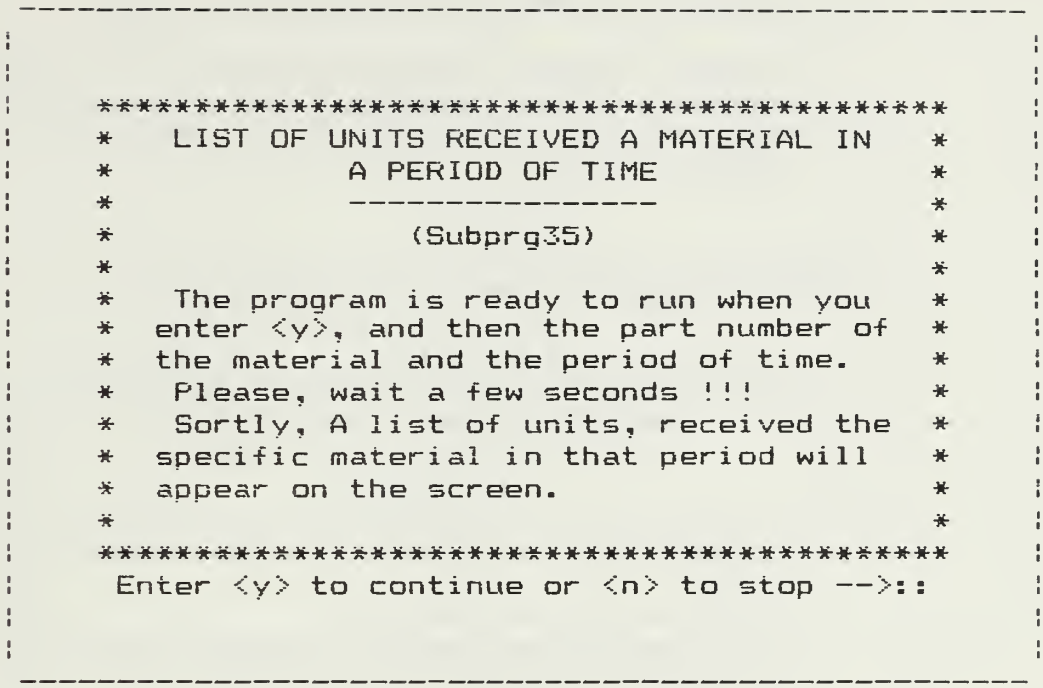


Figure 3.16: Screen for Subprogram35

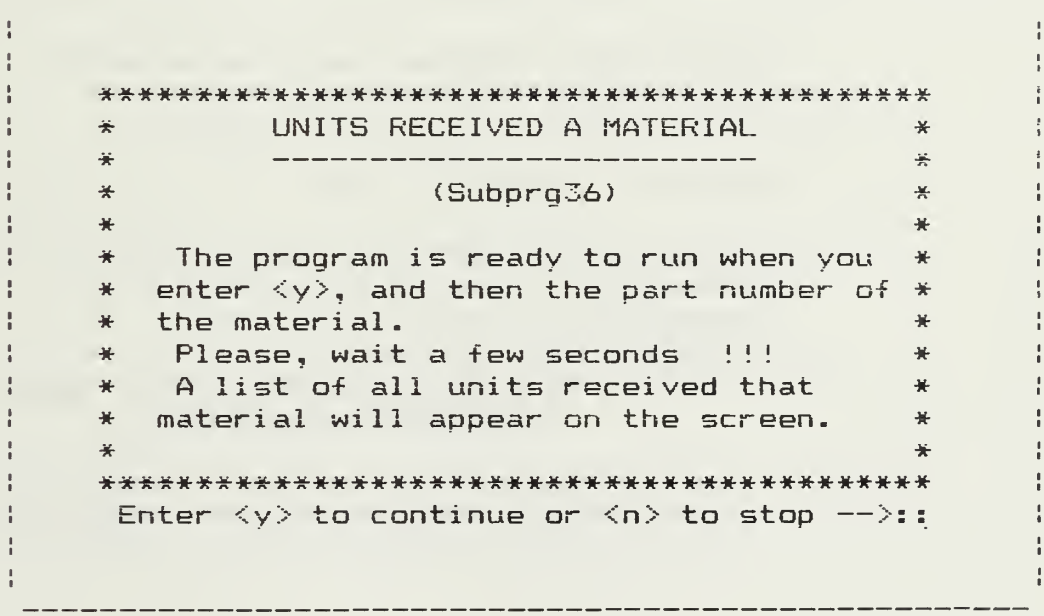


Figure 3.17: Screen for Subprogram36

(g) Screen #15: Materials owed to a unit.

On this screen, the user types a <y>, and the name of the unit. Shortly on the screen will appear the list of all materials owed to that unit. (Figure 3.18)

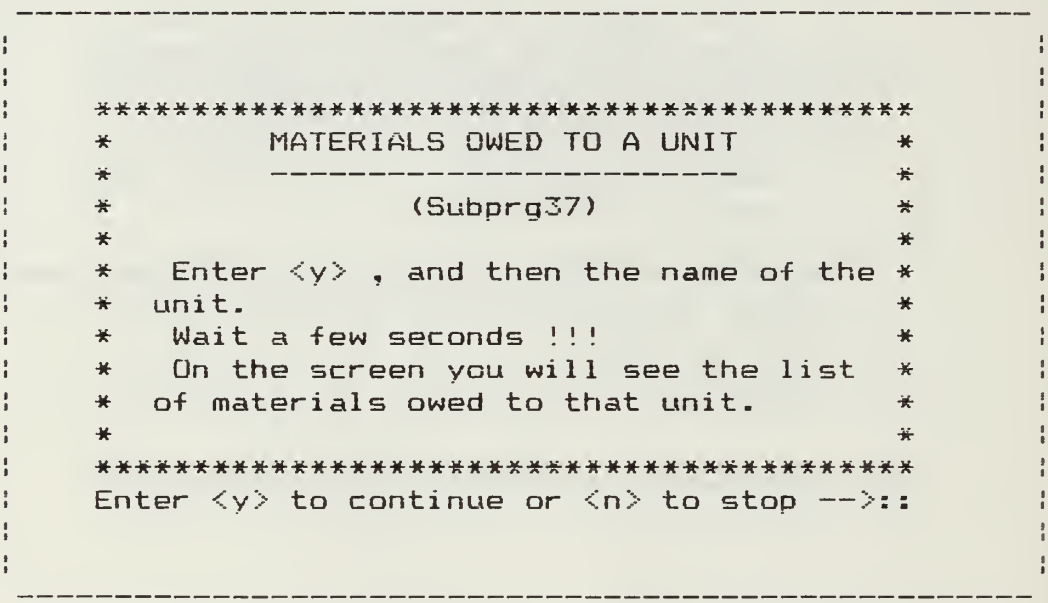


Figure 3.18: Screen for Subprogram37

(h) Screen #16: Units to which a material owed

On this screen there are instructions to the user on how to construct the desired listing. (Figure 3.19)

(i) Screen #17: List of spare-parts needed to repair a material.

According to this screen the user has to type <y> and the name of the main material in order to make the desired listing. (Figure 3.20)

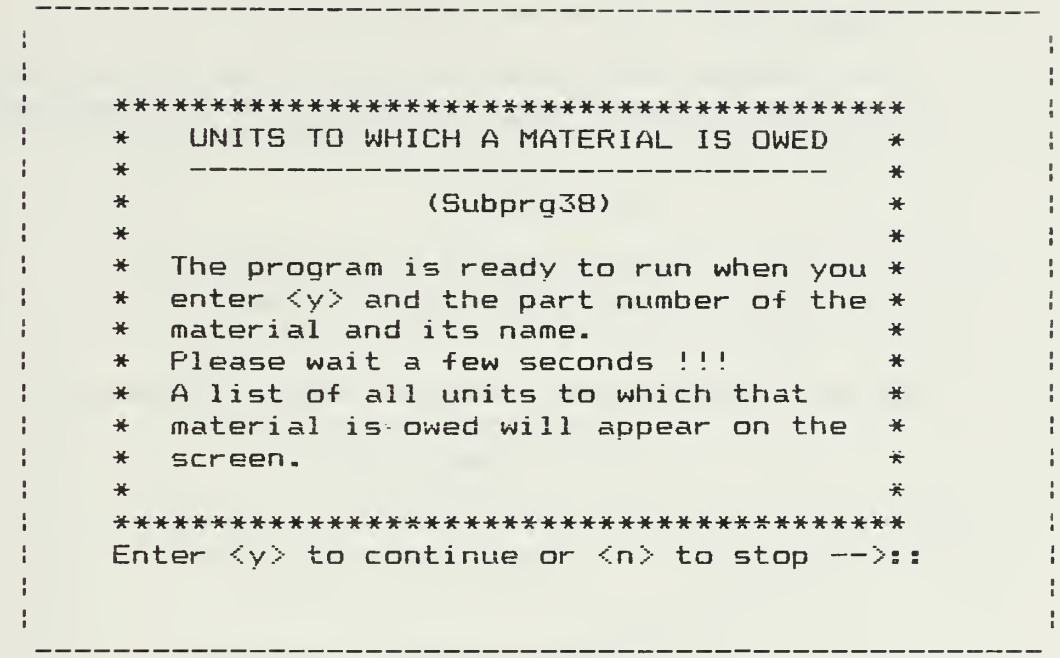


Figure 3.19: Screen for Subprg38

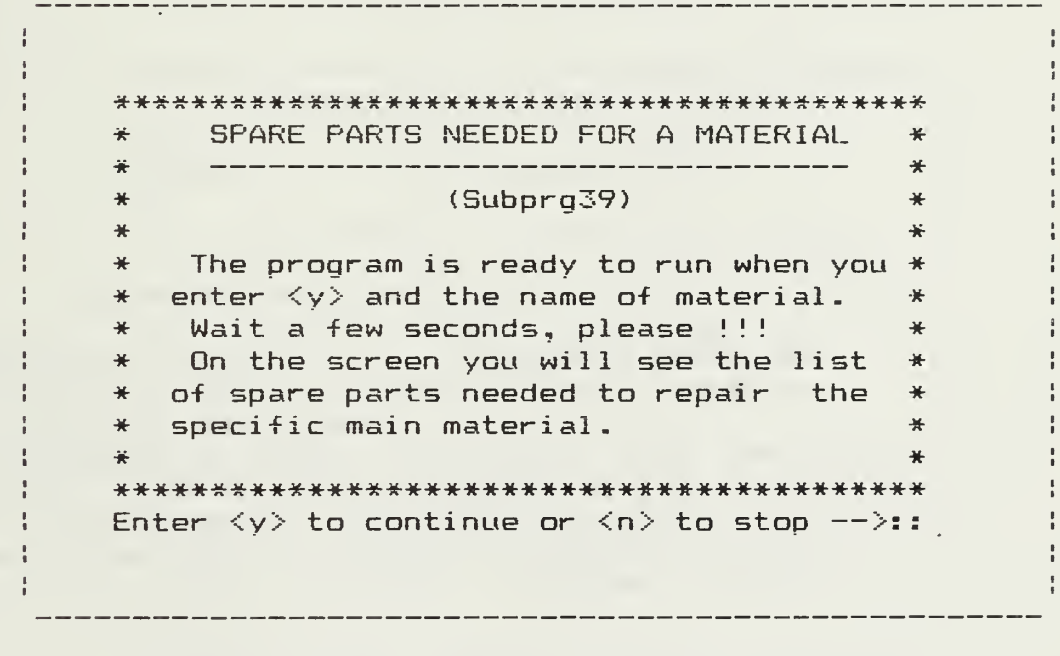


Figure 3.20: Screen for Subprogram39

(6) Screen #18 : Look-ups (Submenu4).

This screen describes the operations of sub-menu4. The user has to type 1 or 2 for the corresponding operations or 0 to exit submenu4. (Figure 3.21)

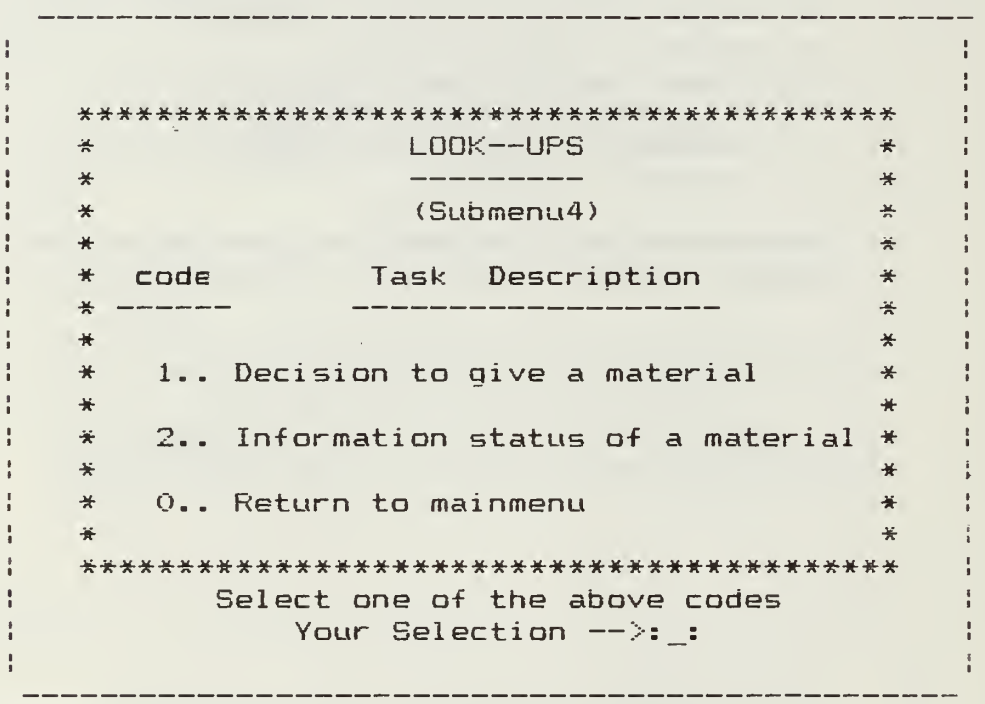


Figure 3.21 : Screen for SUBMENU4.

(a) Screen #19: Decision to give a material

The screen simply gives directions to the user in order to reach the desired results. (Figure 3.22)

(b) Screen #20: Information status of a material.

The user has to follow the instructions of this screen in order to obtain all information about a specific material. (Figure 3.23)

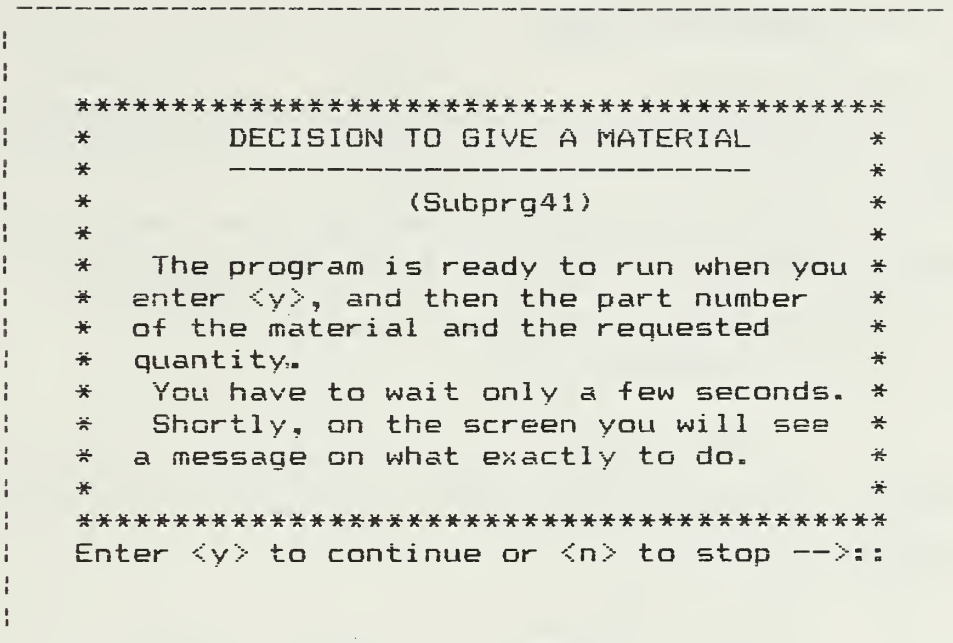


Figure 3.22: Screen for Subprogram41

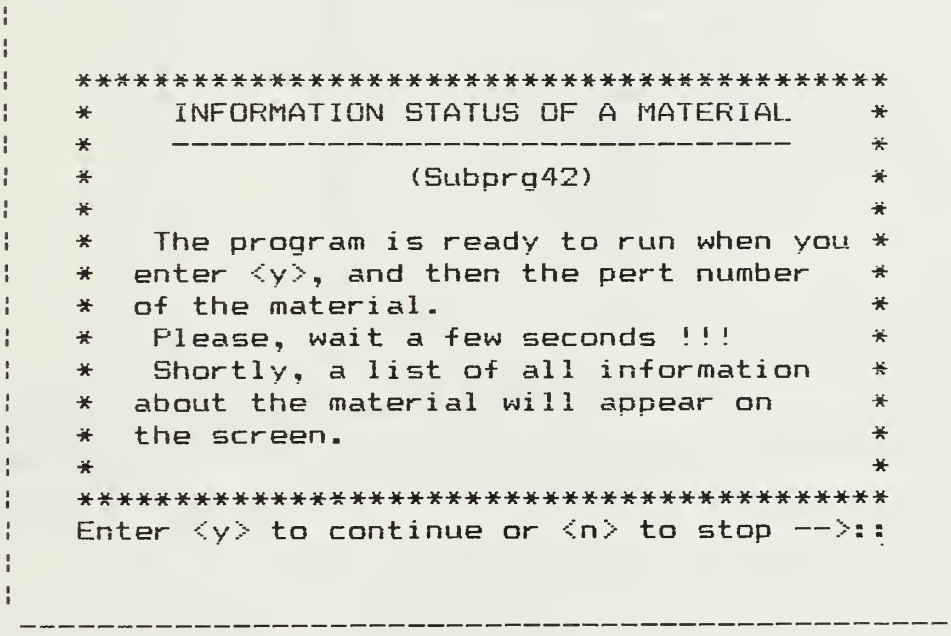


Figure 3.23: Screen for Subprogram42

3. Design of Files

The conceptual view of the system is shown in Figure 3.24.

According to the conceptual view the following files and relations have been specified for the system:

a. INVENTORY File

This file, which is the master file of the system, includes all the materials on hand and has the following fields.

FLD	NAME	TYPE	WIDTH
001	REGENTERNO	N	6
002	REGENTDATE	C	8
003	PARTNO	C	13
004	MATERNAME	C	20
005	QUANTITY	N	5
006	MAX	N	5
007	MIN	N	2
008	INDORDER	N	3
009	MAINMATER	C	15
010	OWED	N	3
011	WAITED	N	4
012	DUEDATE	C	8
013	PLACE	C	8

The field OWED is referring to the materials owed from the Battalion to different units of the Division. The fields WAITED and DUEDATE are referring to the materials owed to Battalion by a hierarchical organ.

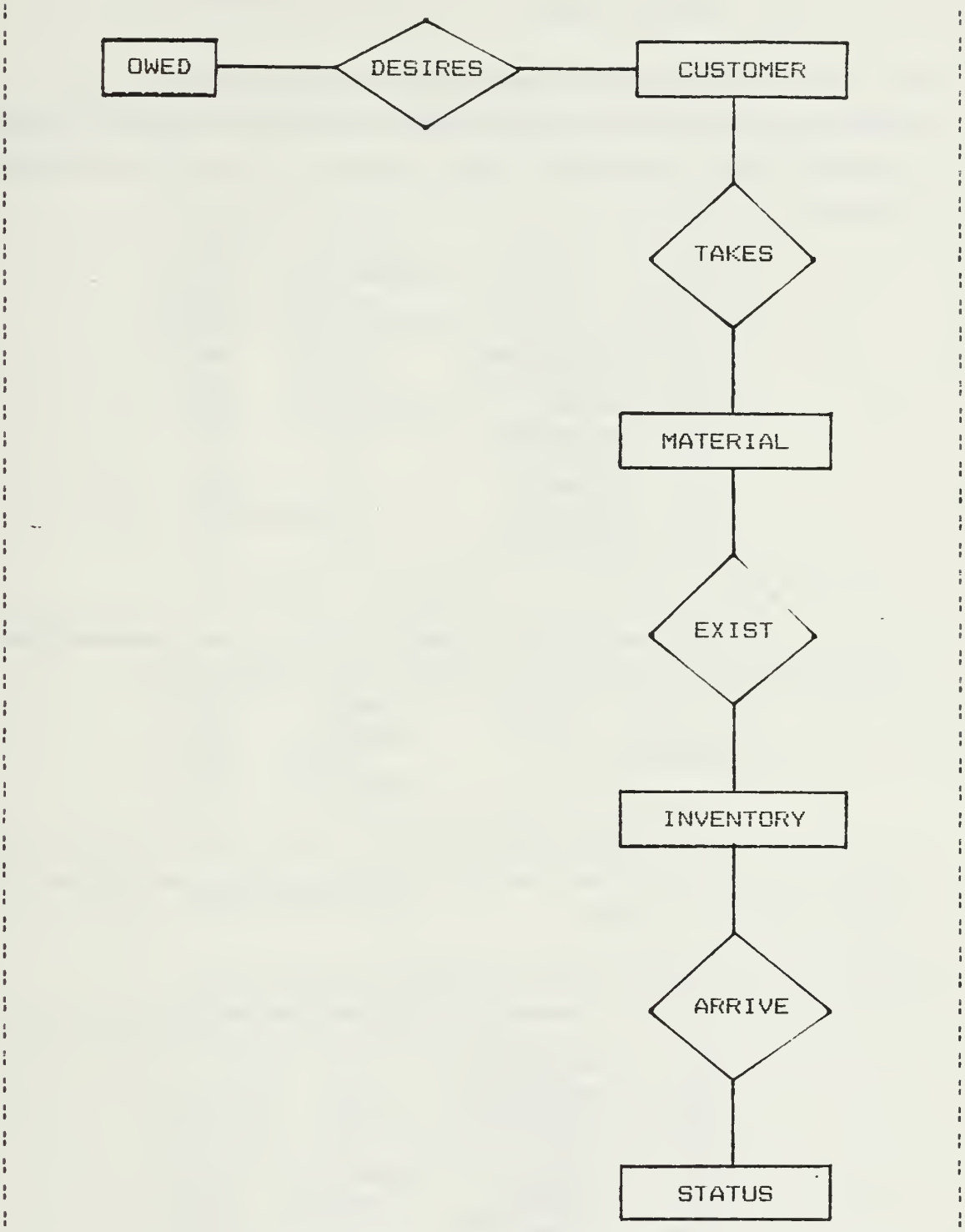


Figure 3.24 : Conceptual View of System.

b. STATUS File

This file contains all materials entering the Battalion in chronological order even if a material has been deleted from the master file. Status file has the following fields:

FLD	NAME	TYPE	WIDTH
001	REGENTERNO	N	6
002	REGENTDATE	C	8
003	PARTNO	C	13
004	MATERNAME	C	20
005	QUANTITY	N	5
006	UNITNAME	C	20
007	REGEXNO	N	6
008	REGEXDATE	C	8

Fields 006-008 identify the link between the material and the supplier organ of the Battalion.

c. OWED File

This file contains the owed materials from the Battalion to different units of the Division and consists of the following fields:

FLD	NAME	TYPE	WIDTH
001	UNITNAME	C	20
002	PARTNO	C	13
003	MATERNAME	C	20
004	QTITY	N	5
005	DUEDATE	C	8
006	REGEXNO	N	6
007	REGEXDATE	C	8

d. MATERIAL File

This file includes the materials that have been given to the units by the Battalion, and has the following fields:

FLD	NAME	TYPE	WIDTH
001	UNITNAME	C	20
002	PARTNO	C	13
003	MATERNAME	C	20
004	QTITY	N	5
005	REGEXNO	N	6
006	REGEXDATE	C	8

e. CUSTOMER File

This file contains all customer units. Its fields are:

FLD	NAME	TYPE	WIDTH
001	UNITNAME	C	20
002	CITY	C	20
003	PHONENO	N	6

f. TAKES Relation

This relation connects the CUSTOMER and MATERIAL files and has the following fields:

FLD	NAME	TYPE	WIDTH
001	UNITNAME	C	20
002	PARTNO	C	13
003	REGEXNO	N	6

g. EXIST Relation

This relation connects the MATERIAL and INVENTORY files and contains the following fields:

FLD	NAME	TYPE	WIDTH
001	PARTNO	C	13
002	REGENTERNO	N	6

h. ARRIVES Relation

This relation connects the INVENTORY and STATUS file containing the following fields:

FLD	NAME	TYPE	WIDTH
001	PARTNO	C	13
002	REGENTERNO	N	6

i. DESIRES Relation

This relation connects the CUSTOMER and OWED files and has the following fields:

FLD	NAME	TYPE	WIDTH
001	UNITNAME	C	20
002	PARTNO	C	13

j. PERSONS File

This file contains the names of all the individuals who are authorized to access the Ordnance Battalion Database.

k. UNPERSON File

In this file the system writes the name of unauthorized individuals trying to enter the database. It has three fields: name,c,15, date,c,8, and time,c,10.

l. STATIC File

This is a statistical file containing the names of all individuals accessing the database. It has three fields: name,c,15, date,c,8, and time,c,10

4. System Functions

The functions of the system are the following:

a. Update of Files

The system allows the user to enter and insert, delete, and modify files. The programs make these changes in all the supporting files. These operations are performed daily.

b. Reorder Processing

The user enters the system and makes a list of all materials which need to be reordered. This function is performed weekly.

c. Report Generators

These functions are for retrieving all the necessary information from the database on a weekly basis, or upon request. Basically, these operations take place in order to produce information reports as shown in Figure 3.1.

d. Look-up Operations

The user can enter the database and take any information about any material as well as ask for a given requested quantity of a material. This function is performed many times daily.

B. PHYSICAL DESIGN.

Physical design contains the activities following the logical design such as production of program software, and working system. Programs must accept input from users, process data, produce reports, and store data in the files.

The supporting programs of the system are as follows:

1. Mainprog with mainmenu are in APPENDIX A.

2. Subprog1 with submenu1 and all the dependent in this program subprograms. That is, subprg11, submen11, subprg12, submen12, subprg13, submen13, are in APPENDIX B.

3. Subprog2 with submenu2, are in APPENDIX C.

4. Subprog3 with submenu3 and all the dependent in this program subprograms. That is, subprg31, submen31, subprg32, submen32, subprg33, submen33, subprg34, submen34, subprg35, submen35, subprg36, submen36, subprg37, submen37, subprg38, submen38, subprg39, and submen39, are in APPENDIX D.

5. Subprog4 with submenu4 and the dependent subprg41 and subprg42, are in APPENDIX E.

IY. IMPLEMENTATION PHASE

Implementation includes all those activities that take place to convert from the old system to the new. The new system may be totally new, replacing an existing manual or automated system, or it may be a major modification to an existing system. In either case, proper implementation is essential to provide a reliable system to meet organization requirements.

In this section we will discuss, for our system, the three aspects of implementation, including training personnel, conversion procedures, and post-implementation review. However the problem of training personnel and conversion planning is beyond the purpose of this thesis and so will be only briefly mentioned.

A. TRAINING

Even well-designed and technically elegant systems succeed or fail because of the way they are operated and used. Those who will be associated with or affected by the system must know in detail what their roles will be, how they may use the system, and what the system will or will not do.

1. Users and Systems Operators Training

In our case the individual is both user and operator.

Users must be instructed first how to operate the equipment. Questions that seem trivial, such as how to turn on a terminal, how to insert a diskette into a micro-computer, or when it is safe to turn off equipment without

danger of data loss, are significant problems to new users who fear computers. Also a user must be able to determine whether a problem arising is caused by the equipment or software or by something they have done in using the system.

Data handling activities must receive the most attention in user training. How to add new data, how to change previously stored data, how to formulate inquiries, and how to delete records of data are questions very significant for the user.

From time to time, users will have to prepare disks, load paper into printers or change ribbons on printers. Users also must be instructed in the method of formatting and testing disks, and copying files.

From the above discussion we conclude that there are two aspects to user training:

a. **Familiarization with the processing system**, that is, the equipment used for data entry or processing, and

b. **Training in using the application**, that is, the software that accepts the data, processes it, and produces the results.

2. Training Methods

The training of users will be in-house, and will be based on manuals that will be prepared for this purpose by the analyst of the new system. The training manuals will be of two approaches. The first, a tutorial, will have the user work through different activities step by step, for example, a check list. The second approach is to create a case study that includes all frequently encountered situations that the system is able to handle and of which the users should be aware. The users must use the system to handle the actual situations.

In the second approach a great deal of time will be spent by the users in making inquiries to retrieve information and records, editing previously entered data, and running reports. Throughout the entire series of activities, troubleshooting activities will be emphasized. Users will become familiar with methods of determining when the system does not perform as expected.

However there is no substitute to experiences. Training manuals are acceptable for familiarization, but the experiences of actually using the equipment, making mistakes, or encountering unexpected situations are the best and most lasting way of learning.

B. CONVERSION

Conversion is the process of changing from the old system to the new one. There are several methods for handling a systems conversion. In our case we will use the method of '**Parallel systems**'.

In this method both systems run in parallel. That is, users continue to operate the old system in the accustomed manner but they also begin using the new system. This method is the safest conversion approach since it guarantees that should problems arise in using the new system, such as errors in processing or inability to handle certain types of transactions, the unit can still fall back to the old system without loss of time. The greatest disadvantage of the parallel systems approach is that the system costs are higher.

1. Conversion Plan

The conversion plan provides a description of all activities that must occur to implement the new system and affect its operation. It identifies the personnel responsible for each activity and includes a timetable of when each activity will occur. During the pre-implementation stages, a list of all tasks containing the following information should be prepared:

a. List of all files that will be used in the new system.

b. List containing all data required to build the above files.

c. List of all documents needed during conversion.

d. Identification of all controls to be used during conversion. Determine how team members will know if something has not been completed properly.

e. Responsibility for each activity.

First someone should be appointed as conversion manager. This individual is the point of contact for outside vendors and for analyst and user personnel. The conversion manager is also responsible for checking all arrangements, reviewing conversion plans, verifying the delivery of equipment, software, forms and whatever is needed for the conversion.

The conversion plan should anticipate possible problems and ways to deal with them. Among the most frequently occuring problems are missing documents, errors in data translation, missing data or lost files, and situations that were overlooked during systems development.

2. Site Preparation

Since the system is a microcomputer, little site preparation work is needed. However, the electrical lines should be checked to ensure they are free of static or power fluctuations. It is a good idea to install a 'clean' line that is not shared by any other equipment. Such machines as electric typewriters and office copiers can interfere with computer operations.

Static electricity is one of the most common foes of computers. Carpet on the floors around computer rooms should be avoided whenever possible since it can create static that in turn is carried by operators. When they touch computer equipment, the static charge can be transferred to the terminal or computer and cause the introduction of errors in the data. If carpet is necessary, it should be the antistatic type that will not allow static buildup.

3. Data and File Preparation

Along with training, the most time consuming aspect of conversion generally involves the preparation of data and systems master file. Since the system is starting from scratch, all necessary data will have to be entered into the system by manual methods. Several thousand material records will have to be keyed into the system from the paper charts. The numbers of personnel who will be assigned this data entry task would be 3 to 4, and it will probably take 6 to 7 weeks to enter all records, about 15,000 to 20,000, into the system. During conversion, it is vital that precautions be included so that no records are overlooked or improperly entered.

a. Record Counts

The most basic control is to ensure that all records are entered in the system. The batch control methods could be used during conversion. Records will be accumulated into groups of 100, each group comprising one batch. During a time period (a morning or afternoon) more than one batch will be entered the system. This method enables the conversion manager to ensure that all records in a batch are entered and that each batch is processed.

At the end of the conversion process, the number of records in the system's master file must equal the number of records in the old systems.

b. Preestablished Totals

In addition to ensuring all records are converted to the new file, the conversion manager must verify that all the field (each record must have the QUANTITY, MAX, MIN, INORDER, OWED, and WAITED quantites) information is valid.

C. POST-IMPLEMENTATION REVIEW

The implementation of the system if well-planned, will go smoothly. When the system has been in use for some time, with virtually no problems have been encountered, analysts and users should jointly review the system after 6 months. The objectives of the this post-implementation review will be to:

1. Determine whether the systems goals and objectives have been achieved.

2. Determine whether user service requirements have been met, while simultaneously reducing errors and costs.

3. Determine whether known or unexpected limitations of the system need attention.

The review is also important to gather information for the maintenance of the system. Since no system is really ever complete, it will be maintained as changes are required because of internal developments, such as new users or business activities, and external developments, such as new requirements etc.

If done properly the new system will be well-received and will meet the performance objectives that led to its development in the first place.

D. IMPLEMENTATION OF DESIGN

1. How to Use the System

After turning on the computer, the user should insert in diskdrive A the diskette of DBASE II and in the diskdrive B the diskette with the project and files. Then he should type ' A>DBASE ' and then ' .SET DEFAULT TO B ' . So the system is ready to run.

The system starts to run by typing from the user ' DO MAINPROG '. In the screen , figure 3.4 apears asking the user to enter the password. If the typed password is not correct the system quits. After typing the correct password by the user the system asks the user to type his last name in order to check in the PERSONS file if he is among the

individuals who are authorized to access the database information. If everything is OK the MAINMENU appears on the screen prompting the user to select the desired SUBMENU by typing the corresponding number from 1 to 4 or <0> to exit the system.

2. Main Menu and Submenus of the System

a. Mainmenu

This program controls the whole operation of the system, and is called from another program called MAINPROG, which is the only program the user calls by name, as we saw in the previous paragraph.

The MAINPROG asks the user to enter the PASSWORD which is for aborting unauthorised users, and then it proceeds by asking the user to enter his name to check if the user is authorized to access the database. Then the MAINPROG proceeds by displaying on the screen the MAINMENU of the system, and pauses waiting for the user to make his choice which is stored in the variable 'MENU'. Then a CASE statement permits the program to branch to the corresponding SUBMENU, or exit the dBASE. If the user selects a wrong choice, then the program gives him an error message, rings the bell, and ask the user to re-enter.

Each time a user enters the database the system automatically writes his name, the date and the time in the STATIC file for statistical purposes only.

If the PASSWORD typed by the user is correct and if his name is in the PERSONS file, he is allowed to continue, otherwise the program automatically exits the dBASE, displaying an appropriate message while at the same time it writes in the UNPERSON file his name, the date and the time.

b. Submenus

All four SUBMENUS operate with the same logic as the mainmenu. Each time, on the screen, appropriate messages appear guiding the user on what to do.

3. Update Operations

These operations permit the user to perform insertions, deletions, and modifications in the files of the database.

a. Insertion Operation

This operation is performed by the SUBPRG11 program. This program, actually, is for adding a quantity of materials in the INVENTORY file and modifying the other fields of the record of that file. It also adds a new record in the INVENTORY and STATUS files whenever it is necessary.

The whole structure of the program is as follows

(1) The INVENTORY file is opened.

(2) The user is prompted to enter:

(a) The part number of the material.

(b) The name of the material.

(c) The received quantity.

(d) The owed quantity.

(e) The waited quantity.

(f) The registration number, and

(g) The date.

(3) Then the program searches the INVENTORY file for the part number of the material. If it does not exist, it automatically creates a new record in both INVENTORY and STATUS files after asking the user the following complement information about the received material:

(a) The maximum quantity.

(b) The minimum quantity.

(c) The index for reorder.

(d) The main material, and

(e) The place in the store.

Otherwise, it proceeds by checking the material name. Is the material names are different the program stops and a message appears on the screen, otherwise the program proceeds making all the necessary changes in the fields of the INVENTORY file. Then it displays the old and the updated records of the specific material.

b. Deletion Operation

This operation is performed by the SUBPRG12 program. This program, actually, is for subtracting a quantity of materials from the INVENTORY file , and adding a new record of that material in the MATERIAL and OWED files whenever it is necessary.

The whole structure of that program is as follows:

(1) The INVENTORY file is opened.

(2) The user is prompted to enter:

(a) The part number of the given material.

(b) The material name.

(c) The given quantity.

(d) The owed quantity.

(e) The due date.

(f) The registration number, and

(g) The date.

(3) The program searches the INVENTORY file for the part number of the material. Then, after locating the record of the material, it subtracts the given quantity from INVENTORY file, it adds a new record in the MATERIAL file with the given quantity, and if there is an owed quantity it adds a new record in the OWED file.

c. Modification Operation

This operation is performed by the SUBPRG13 program. This program changes the name of a material.

The structure of the program is as follows:

(1) The INVENTORY file is opened.

(2) The user is prompted to enter if he wants to proceed or not. If the user enters <n> the program returns to SUBMENU1, otherwise it asks the user to enter the old name of the material.

(3) Then the program searches the INVENTORY file for the material name. If the material does not exist in the file an appropriate message appears on the screen and the process quits, otherwise it asks the user to enter the new name of the material. Then the program changes the old name with the new one in the INVENTORY file and at the same time it changes the names in STATUS and OWED files.

(4) The whole operation continues under the control of a WHILE loop, until the user exists.

3. Reorder Operation

This operation is performed by the SUBPROG2 program. This program prints a list of materials needing to be reordered.

The whole operation of this program is as follows:

a. The INVENTORY file is opened.

b. The user is prompted to enter <y> if he wants to proceed the program, or <n> otherwise.

c. If the user enter <n> the program returns to the mainmenu. In the contrary it continues as follows:

   (1) It creates from INVENTORY file a temporary file called tmp1, which contains all materials the quantity of which is under the index-order.

   (2) Then it creates from tmp1 file the ORDER file which contains all materials needing to be reordered. For this operation the program considers the quantity on hands of the material, the awaited quantity, and the owed quantity.

d. Finally it prints a list containing all materials needing to be reordered.

4. Report Generator Operations

These programs permit the user to print all the necessary reports which must be available to any responsible individual for decision making purposes. These program reports are as follows:

a. List of Materials Received on a Particular DATE

This function is performed by SUBPRG31 program. This program prints a list of all materials received by the Ordnance Battalion on a particular date.

The program uses the following structure:

(1) The user is prompted to enter <y> to proceed with the function, or <n> to return to the SUBMENU3.

(2) The system asks the user to enter the particular DATE.

(3) The necessary heading of the report is printed.

(4) The INVENTORY file is opened, and the program copies to a temporary file called tmp1 all records of materials with the same registration date as that entered by the user.

(5) The program prints the list of the materials in date order into the tmp1 file.

In case of error an appropriate message appears on the screen.

b. List of Materials Received in a Period of Time

This function is performed by SUBPRG32 program. This program prints a list of materials received by the Ordnance Battalion in a specific period of time.

The structure of this program is exactly as the previous one with the only difference being it uses a PERIOD of TIME instead of TIME.

c. Materials Received by a REGISTRATION NUMBER

This function is performed by SUBPRG33 program. This program prints a list of materials received by the Ordnance Battalion by a specific Registration Number.

The structure of this program is the same as the previous programs so its description is not needed. The only difference is that the user enters the registration number as data selection criteria.

d. List of Materials Given on a Specific DATE or PERIOD of TIME

This function is performed by SUBPRG34 program. This program prints a list containing all materials given by the Ordnance Battalion to any Unit of the Division either on a particular DATE or in a specific PERIOD of TIME.

The structure of the program is as follows:

(1) The user is prompted to enter <y> to proceed with the program or <n> to return to the SUBMENU3.

(2) The system asks the user to select option (number 1 for a specific DATE or number 2 for a PERIOD of TIME), and then enter the desired DATE or PERIOD of TIME correspondingly.

(3) The MATERIAL file is opened, and the program copies to a temporary file all materials given to the units of the Division on that DATE or PERIOD of TIME.

(4) Then it uses the temporary file to print the list with all materials given by the Battalion to the Units of the Division.

e. List of units received a specific material in a PERIOD of TIME

This function is performed by SUBPRG35 program. This program prints a list of units of the Division to which a specific material was given by the Battalion in a particular period of time.

The structure of the program is as follows:

(1) The user is prompted to enter:
   (a) <y> if he wants the program to proceed or <n> otherwise.
   (b) The part number of the material.
   (c) The start and end date

(2) Then the necessary heading of the report is printed.

(3) The MATERIAL file is opened, and the program copies to a temporary file called tmp1, all units from the MATERIAL file all units receiving the specific material in that particular period of time.

(4) It uses tmp1 file to create a sorted file called P1 according to registration date.

(5) It uses the P1 file to print the desired list of Units.

(6) In each case an appropriate message appears on the screen guiding the user.

f. List of Units Received a Material

This function is performed by SUBPRG36 program. This program prints a list of units of the Division to which a specific material was given by the Battalion.

The structure of the program is exactly the same as the previous program so it is not necessary to describe it. The only difference is that there is not need for the user to enter the period of time.

g. List of Materials Owed to a Unit

This function is performed by the SUBPRG37 program. This program prints a list of materials owed by the Battalion to a specific unit of the Division.

The structure of the program is about the same as the two previous programs except that the user enters the name of the specific unit, where upon the program searches the OWED file instead of MATERIAL file. Finally a list of materials owed to the specific unit by the Battalion appears on the screen.

h. List of Units to which a Specific Material is Owed

This function is performed by the SUBPRG38 program. This program prints a list of Units of the Division to which a specific material is owed by the Ordnance Battalion.

The structure of the program is as follows:

(1) The user is prompted to enter <y> to proceed or <n> to return to SUBMENU3.

(2) Then the user has to type the part number and the name of the specific material.

(3) The OWED file is opened, and the program copies to tmp1 file from the OWED file all units to which the specific material is owed.

(4) It uses the tmp1 file to print the desired list of Units.

i. List of Spare-Parts Needed to Repair a Material

This function is performed by the SUBPRG39 program. This program prints a list containing all spare parts needed to repair a main material.

The structure of the program is as follows:

(1) The user is prompted to enter <y> to continue the process or <n> to return to SUBMENU3, and then the name of the main material.

(2) The INVENTORY file is opened and the program proceeds by copying to tmp1 file all spare parts needed to repair the specific main material.

(3) Finally it uses the tmp1 file to print the desired list.

5. Looks-ups Operations

These operations are to help the user to obtain information about the status of a specific material or for decision making purposes. These oprations are the following:

a. Decision to Give a Material

This function is performed by the SUBPRG41 program. This program gives a response to the user on what to do in the case of a request of a material by a Unit of the Division.

The structure of the program is as follows:

(1) The user is prompted to enter <y> to continue the process or <n> to return to SUBMENU4, and then the part number of the material and the requested quantity.

(2) The necessary memory variables are initialized.

(3) The INVENTORY file is opened and the program proceeds to locate the specific material. In case that the material does not exist in the file, an appropriate message appear on the screen. Otherwise it examines the existing quantity and a series of messages print out, guiding the user on what to do, as for example 'you can give the requested material' or ' You can give only that quantity of the material' or ' I'm sorry you can not give any quantity of that material' or ' Go to your boss to decide what to do'.

In particular, the program does the following:

(a) Permits the user to give the requested quantity of materials or.

(b) Tells him that the material does not exist in the Battalion and he must place an order or,

(c) Informs him that the material exists in the Battalion but in a limited quantity, and he must consult with the Commander who will decide what to do.

b. Information Status of a Material

This function is performed by the SUBPRG42 program. This program gives the user the following information about a specific material:

(1) On hands quantity.
(2) Maximum quantity
(3) Minimum quantity
(4) Waited quantity
(5) Owed quantity to the Battalion
(6) Owed quantity to units
(7) Given quantity to units

(8) Received quantity from the Battalion

The structure of the program is as follows:

(1) The user is prompted to enter <y> to continue or <n> to return to submenu4, and then the part number of the material.

(2) The Inventory, Owed, and Status files are opened and the program proceeds locating the above information.

(3) Finally it displays all the above information.

V. CONCLUSIONS AND RECOMMENDATIONS

The previous pages describe a material database system model, suitable for implementation within the Hellenic Armed Forces Formations, primarily for an Ordnance Battalion of an Infantry Division. This system could also be applied to any Ordnance unit, at higher or lower levels than a Battalion with only slight modification.

Section I was dedicated to introducing several general database concepts. Section II supports the analysis phase, section III presents the design phase, and section IV contains the implementation of the system.

The goal of the system is to achieve greater processing speed, better accuracy and consistency, faster information retrieval, reduced cost, security, and better use of personnel. The Commander of the Battalion will be able to make faster decisions concerning the material, which in itself is of prime importance.

DBASE II was used as a database management system, since it is readily available based on the relational model, thereby increasing independency, and reduces redundancy. In addition, dBASE II contains its own programming language, which is a structured, high level language, generally very efficient for manipulating data in the database. Computing services will be provided by a microcomputer, IBM PC or compatible with two floppy disks and a hard disk of a size dependent upon the volume of data processed.

The designed functions of the system are: Update files, Reorder processing, Report generators, and Look-ups. The functions most usually needed, have been implemented, but a wide variety of other functions could also be created.

The software life cycle has been taken into account during the program development process. Programs are easy to modify to meet future improvement needs. A top-down design approach has been used in this implementation which serves the the goal of the system.

The implementation phase contains the implementation of design and briefly the training of personnel and conversion planning. In this implementation only the main categories of materials have been used, with a limited and amount of data concerning each of them. Future improvements could include the entire military materials listing, with more complete information about the records.

This thesis constitutes a good basis for future computerization of the military materials in the Hellenic Armed Forces Formations.

APPENDIX A

Appendix A contains the main program (MAINPROG) and the main menu (MAINMENU) as follows:

1. Main Program (MAINPROG)

```
ERASE
SET TALK OFF
@  3,10 SAY 'O r d n a n c e   B a t t a l i o n ,
            D a t a b a s e'
@  5,20 SAY 'S e c u r i t y   S y s t e m'
STORE DATE()  TO dt
STORE '       '  TO password
@ 10,26 SAY ' Enter  Password: '
SET console off
ACCEPT  TO password
SET console on
SET EXACT ON
IF password = 'panos'
 SET EXACT OFF
 STORE '                    '  TO LN
 @  12,20 SAY 'Enter your Last Name --->'  GET LN
 READ
 use persons
 LOCATE FOR name = LN
 IF  EOF
    @  14,20 SAY 'I'm sorry , you are unauthorized,
                to  use  the Database'
    use unperson
    APPEND BLANK
    REPLACE  name WITH  LN, date WITH dt
    quit
 ELSE
    STORE  '  :  :     '  TO tm
    @ 14,28 SAY 'Enter time --->'  GET tm PICTURE ,
                '99:99:99AA'
    READ
    use static
    APPEND BLANK
    REPLACE  name WITH LN, date WITH dt, time WITH tm
    STORE  F   TO FINISHED
    DO WHILE .NOT. FINISHED
       ERASE
       DO MAINMENU
       STORE ' ' TO MENU
       @ 24,10 SAY 'Your selection -->' GET MENU
       READ
```

```
        DO WHILE .NOT. MENU $ '12340'
           @ 24,30 SAY '<<=== Invalid Choice.  Please,
                        Re-enter.'
           STORE ' ' TO MENU
           @ 24,10 SAY 'Your Selection -->' GET MENU
           READ
        ENDDO
        @ 24,30
        DO CASE
          CASE MENU = '1'
            DO subprog1
          CASE MENU = '2'
            DO subprog2
          CASE MENU = '3'
            DO subprog3
          CASE MENU = '4'
            DO subprog4
          CASE MENU = '0'
            STORE  T  TO FINISHED
        ENDCASE
     ENDDO?
     RELEASE MENU
 ENDIF
ELSE
 ?' Unauthorized Person'
 quit
ENDIF
ERASE
SET TALK ON
RETURN
```

2. Main Menu (MAINMENU)

```
ERASE
@  1,60 SAY DATE()
@  1,15 SAY '*******************************'
@  2,15 SAY '*                             *'
@  3,15 SAY '*     M A I N    M E N U      *'
@  4,15 SAY '*                             *'
@  5,15 SAY '*******************************'
@  7,15 SAY 'Select one of the following '
@  9,10 SAY '.....................................'
@ 11,10 SAY '.  1...     Update of files         .'
@ 13,10 SAY '.  2...     Reorder Processing      .'
@ 15,10 SAY '.  3...     Report Generator        .'
@ 17,10 SAY '.  4...     Looks-ups               .'
@ 19,10 SAY '.  0...     Exit the system         .'
@ 21,10 SAY '.....................................'
RETURN
```

APPENDIX B

Appendix B contains Subprogram1 (SUBPROG1) and Submenu1 (SUBMENU1) with all the dependent subprograms: SUBPRG11, SUBMEN11, SUBPRG12, SUBMEN12, SUBPRG13, and SUBMEN13.

1. Subprogram1 (SUBPROG1)

```
ERASE
SET TALK OFF
DO secure
STORE  F  TO FINISH1
DO WHILE .NOT. FINISH1
   DO submenu1
   STORE ' ' TO choise1
   @ 23,10 SAY 'Enter your selection --->' GET choise1
   READ
   DO WHILE .NOT. choise1 $ '0123'
      @ 23,38 SAY '<<=== Invalid Choice.Re-enter'
      STORE ' ' TO choise1
      @ 23,10 SAY 'Enter your selection --->' GET choise1
      READ
   ENDDO
   DO  CASE
       CASE  choise1 = '1'
             DO   subprg11
       CASE  choise1 = '2'
             DO   subprg12
       CASE  choise1 = '3'
             DO   subprg13
       CASE  choise1 = '0'
             STORE  T  TO  FINISH1
   ENDCASE
ENDDO
RELEASE FINISH1
SET TALK ON
RETURN
```

2. Submenu1 (SUBMENU1)

```
ERASE
@  1,10 SAY DATE()
@  2,10 SAY '*************************************************'
@  3,10 SAY '*                                               *'
@  4,10 SAY '*            UPDATE  FILE  FUNCTIONS            *'
@  5,10 SAY '*            ------------------------           *'
@  6,10 SAY '*                  (Submenu1)                   *'
@  7,10 SAY '*                                               *'
@  8,10 SAY '* code          Task  Description               *'
@ 09,10 SAY '* ----          ----------------                 *'
@ 10,10 SAY '*                                               *'
@ 11,10 SAY '*  1..  Insert a material's quantity            *'
@ 12,10 SAY '*                                               *'
@ 13,10 SAY '*  2..  Delete a material's quantity            *'
@ 14,10 SAY '*                                               *'
@ 15,10 SAY '*  3..  Modify a material's name.               *'
@ 16,10 SAY '*                                               *'
@ 17,10 SAY '*  0..  Return to mainmenu.                     *'
@ 18,10 SAY '*                                               *'
@ 19,10 SAY '*************************************************'
@ 21,20 SAY 'Select one of the above codes'
RETURN
```

3. Subprogram11 (SUBPRG11)

```
ERASE
SET TALK OFF
DO submen11
STORE ' ' TO continue
@ 22,09 SAY 'Enter <y> to continue or <n> to stop' GET ,
            continue
READ
DO WHILE continue <> 'n'
   ERASE
   IF continue = 'y'
      STORE  '                    '        TO  pn
      STORE  '                         ' TO  mn
      STORE  '      '                    TO  qt
      STORE  '      '                    TO  od
      STORE  '          '                TO  ren
      STORE  '          '                TO  red
      @  5,10 SAY 'Part number       :'  GET pn ,
                  PICTURE '9999-999-9999'
      @  6,10 SAY 'Material name     :'  GET mn
      @  7,10 SAY 'Received quatity :' GET qt PICTURE '99999'
      @  8,10 SAY 'Owed quantity     :' GET od PICTURE '99999'
```

```
@  9,10 SAY 'Reg. enter number:' GET ren ,
              PICTURE '99999'
@ 10,10 SAY 'Reg. enter date  :' GET red ,
              PICTURE '99/99/99'
READ
STORE   0  TO  Q
STORE   0  TO  QQ
STORE   0  TO  D
STORE   0  TO  NUM
STORE val(ren) TO NUM
STORE val(qt)  TO Q
STORE val(od)  TO D
use inventory
LOCATE FOR  partno = pn
IF EOF
   STORE '      '              TO  mx
   STORE '   '                 TO  mina
   STORE '    '                TO  io
   STORE '                  '  TO  mm
   STORE '     '               TO  wt
   STORE '          '          TO  pl
   @ 12,10 SAY '* Record does not exist *'
   @ 14,10 SAY 'Max                :' GET mx ,
               PICTURE '99999'
   @ 15,10 SAY 'Min                :' GET mina ,
               PICTURE '99'
   @ 16,10 SAY 'Index Order        :' GET io ,
               PICTURE '999'
   @ 17,10 SAY 'Main material      :' GET mm
   @ 18,10 SAY 'Waited             :' GET wt ,
               PICTURE '9999'
   @ 19,10 SAY 'Place              :' GET pl
   READ
   STORE val(mx)     TO MAXQ
   STORE val(mina)   TO MINQ
   STORE val(io)     TO IORD
   STORE val(wt)     TO WQ
   APPEND BLANK
   REPLACE regenterno with NUM, regentdate with red,
     partno with pn, matername with mn, max with MAXQ,
     quantity with Q, min with MINQ, owed with D
   REPLACE indorder with IORD, mainmater with mm,
     waited with WQ, place with pl
   ?
   ?
   LOCATE FOR partno = pn
   ?' The new Record is:'
   ?
   DISPLAY
   ?
ELSE
```

```
         ?' The old Record is:'
         ?
         ?
         DISPLAY
         ?
         ?
         IF  maternname = mn
            STORE  Q + quantity  TO  QQ
            IF  waited = Q  .AND. D = O
                REPLACE  quantity with QQ, waited with O ,
                    regenterno with NUM, regentdate with red
            ELSE
                REPLACE  quantity with QQ, waited with D ,
                    regenterno with NUM, regentdate with red
            ENDIF
            ?
            ?
            LOCATE FOR partno = pn
            ?'The new update Record is:'
            ?
            ?
            DISPLAY
            ?
            ?
         ELSE
            ?
            ?' Maternname has been changed. DO first subprg13'
            ?
         ENDIF
      ENDIF
   ELSE
      ?
      ?
      ?'     **     Incorrect  argument     **'
      ?
   ENDIF
   ?
   ?
   ?
   ?
   ?
   ?
   ?
   STORE ' ' TO continue
   @ 24,10 SAY 'enter  <y> to continue  or <n> to stop' ,
               GET continue
   READ
   ?
ENDDO
SET TALK ON
RETURN
```

4. Submenu11 (SUBMEN11)

```
ERASE
?
?'*************************************************************'
?'*                                                           *'
?'*              INSERT A MATERIAL'S QUANTITY                 *'
?'*              ----------------------------                 *'
?'*                        (Subprg11)                         *'
?'*                                                           *'
?'*      The program is ready to run when you enter           *'
?'*                                                           *'
?'*      <y> and then the part number of material,            *'
?'*                                                           *'
?'*      the material name, and the received quantity.        *'
?'*                                                           *'
?'*      Please, wait a few seconds and the old and           *'
?'*                                                           *'
?'*      updated record of the material will appear           *'
?'*                                                           *'
?'*      on the screen.                                       *'
?'*                                                           *'
?'*************************************************************'
RETURN
```

5. Subprogram12 (SUBPRG12)

```
ERASE
SET TALK OFF
DO submen12
STORE ' ' TO continue
@ 22,10 SAY 'enter <y> to run or <n> to stop ==>' ,
            GET continue
READ
DO WHILE continue <> 'n'
   ERASE
   IF continue = 'y'
      STORE  '                '       TO  pn
        STORE '                        ' TO mn
        STORE '      '                   TO qt
        STORE '      '                   TO od
        STORE '         '                TO dd
        STORE '       '                  TO ren
        STORE '          '               TO red
        STORE '                         ' TO un
        @  6,10 SAY 'Part number      :' GET pn ,
                    PICTURE '9999/999/9999'
        @  7,10 SAY 'Material name    :' GET mn
```

```
@  8,10 SAY 'Quantity          :' GET qt ,
                PICTURE '99999'
@  9,10 SAY 'Owed              :' GET od ,
                PICTURE '99999'
@ 10,10 SAY 'Due date          :' GET dd ,
                PICTURE '99/99/99'
@ 11,10 SAY 'Reg. Exit No      :' GET ren ,
                PICTURE '999999'
@ 12,10 SAY 'Reg. Exit Date    :' GET red ,
                PICTURE '99/99/99'
@ 13,10 SAY 'Unit name         :' GET un
READ
STORE   0  TO  Q
STORE   0  TO  QQ
STORE   0  TO  D
STORE   0  TO  NUM
STORE  val(qt)   TO  Q
STORE  val(od)   TO  D
STORE  val(ren)  TO  NUM
use inventory
LOCATE FOR partno = pn
IF  EOF
  @ 15,10 SAY 'Requested material does not exist ,
               in the Battalion'
ELSE  .
 IF D = 0
    STORE  quantity - Q  TO  quantity
    use material
    @ 15,10 SAY 'Create new record in MATERIAL file'
    APPEND BLANK
    REPLACE unitname with un, partno with pn
    REPLACE matername with mn, qtity with Q
    REPLACE regexno with NUM, regexdata with red
    ?
    DISPLAY
 ELSE
    IF Q = 0
       use owed
       @ 15,10 SAY 'Create new record in OWED file'
       APPEND BLANK
       REPLACE unitname with un, partno with pn,
          matername with mn, qtity with D,
          duedate with dd, regexno with NUM
       REPLACE regexdate with red
    ELSE
       use inventory
       LOCATE  FOR  partno = pn
       STORE  quantity - Q  TO  quantity
       use  owed
       @ 15,10 SAY 'Create new record in OWED file'
       APPEND BLANK
```

```
                REPLACE unitname with un, partno with pn
                REPLACE maternname with mn, qtity with D
                REPLACE duedate with dd, regexno with NUM
                REPLACE regexdate with red
                use material
                @ 17,10 SAY 'Create new record in MATERIAL .
                           file'
                APPEND BLANK
                REPLACE unitname with un, partno with pn
                REPLACE maternname with mn, qtity with Q
                REPLACE regexno with NUM, regexdate with red
             ENDIF
          ENDIF
       ENDIF
    ELSE
       ?'     **     Incorrect  argument     **'
    ENDIF
    ?
    STORE ' ' TO continue
    @ 22,10 SAY 'Enter <y> to continue or <n> to stop -->' .
    ~           '   GET continue
    READ
ENDDO
ERASE
SET TALK ON
RETURN
```

6. Submenu12 (SUBMEN12)

```
ERASE
?'*************************************************************'
?'*                                                           *'
?'*            DELETE A MATERIAL'S QUANTITY                   *'
?'*            ----------------------------                   *'
?'*                      (Subprg12)                           *'
?'*                                                           *'
?'*     The program is ready to run when you enter            *'
?'*                                                           *'
?'*     <y>, and then the part number of material,            *'
?'*                                                           *'
?'*     the material name, the given quantity, and            *'
?'*                                                           *'
?'*     the unit name.                                        *'
?'*                                                           *'
?'*     Please, wait a few seconds and the old and            *'
?'*                                                           *'
?'*     updated records will appear on the screen.            *'
?'*                                                           *'
?'*************************************************************'
RETURN
```

7. Subprogram13 (SUBPRG13)

```
ERASE
SET TALK OFF
DO submen13
STORE ' ' TO change
@ 18,01 SAY 'enter <y> to change the material name or <n> ,
            to stop -->'  GET change
READ
DO WHILE change <> 'n'
   ERASE
   use inventory
   if change = 'y'
      STORE '                         '  TO  oldname
      @ 10,15 SAY 'Enter old name ==>'  GET oldname
      READ
      LOCATE  FOR  maternname = oldname
      if  .NOT.  EOF
          STORE '                         '  TO newname
          @ 12,15 SAY 'Enter new name -->'  GET newname
          READ
          REPLACE  maternname WITH newname
          ?
          ?' In  INVENTORY  file :'
          ?' -------------------'
          DISPLAY
          ?
          ?' In  OWED  file :'
          ?' --------------'
          use owed
              REPLACE ALL maternname WITH newname ,
                      FOR maternname = oldname
              DISPLAY  ALL  FOR   maternname = newname
              ?
              ?' In  MATERIAL  file :'
              ?' ------------------'
          use material
              REPLACE  ALL maternname WITH newname ,
                       FOR maternname = oldname
              DISPLAY  ALL  FOR   maternname = newname
              ?
              ?
      ELSE
        @ 15,15 SAY 'This record does not exist in the files'
      ENDIF
   ELSE
      ?
      ?' **  Incorrect  argument  **'
   ENDIF
   ?
```

```
   ?
   ?
   ?
   STORE ' ' TO change
   @ 22,15 SAY 'enter <y> to continue or <n> to stop',
                GET change
   READ
ENDDO
ERASE
SET TALK ON
RETURN
```

8. Submenu13 (SUBMEN13)

```
ERASE
?'*************************************************************'
?'*                                                           *'
?'*              MODIFY MATERIAL'S NAME                       *'
?'*              ----------------------                       *'
?'*                     (Subprg13)                            *'
?'*                                                           *'
?'*   This program interactively changes a material name      *'
?'*   into a new one.  The only thing to do is to type        *'
?'*   <y>, and then the old and the new name, and in a        *'
?'*   few seconds the old and the new records of the          *'
?'*   material appears on the screen.                         *'
?'*                                                           *'
?'*   The modification is applied into all used in the        *'
?'*   database files of INVENTORY, OWED, and MATERIAL.        *'
?'*                                                           *'
?'*************************************************************'
RETURN
```

APPENDIX C

Appendix C contains the subprogram2 (SUBPROG2) and the submenu2 (SUBMENU2) as follows:

1. Subprogram2 (SUBPROG2)

```
ERASE
SET TALK OFF
DO submenu2
STORE ' ' TO continue
@ 18,07 SAY 'Enter <y> to continue or <n> to stop -->' ,
            GET continue
READ
DO WHILE continue <> 'n'
   ERASE
   IF continue = 'y'
      @ 10,12 SAY 'T h e  p r o g r a m  i s  r u n n i n g'
      @ 12,10 SAY 'Please.  Wait a few minutes to take,
                   the results'
      USE INVENTORY
      COPY to tmp1 FIELD PARTNO,MATERNAME,QUANTITY,MAX,,
             INDORDER,OWED,WAITED FOR (quantity <= indorder)
      SELECT PRIMARY
      USE tmp1
      DO WHILE .NOT. EOF
         STORE  '                ' to P
         STORE  '                      ' to MN
         STORE  0  to Q
         STORE  0  to M
         STORE  0  to IN
         STORE  0  to O
         STORE  0  to W
         STORE  0  to QW
         STORE  0  to QWO
         STORE  0  to OQW
         STORE  0  to ORDR
         STORE  PARTNO       to P
         STORE  MATERNAME    to MN
         STORE  QUANTITY     to Q
         STORE  MAX          to M
         STORE  INDORDER     to IN
         STORE  OWED         to O
         STORE  WAITED       to W
         STORE  Q + W        to QW
         IF  QW > O
             STORE  QW-O     to QWO
```

```
        IF  QWO < IN
            STORE  M-QWO to ORDR
        ENDIF
    ELSE
       STORE  O-QW      to OQW
       STORE  M+OQW     to ORDR
    ENDIF
    IF  ORDR > 0
        SELECT  SECONDARY
        USE  ORDER
        APPEND  BLANK
        REPLACE  PARTNO WITH P, MATERNAME WITH MN
        REPLACE  QTITY WITH  ORDR
    ENDIF
    SELECT  PRIMARY
    SKIP
ENDDO WHILE
STORE  'LIST'  TO title1
STORE  '----'  TO under11
STORE  'of all materials needed reoeder' TO title2
STORE  '-----------------------------' TO under12
SELECT  PRIMARY
USE  ORDER
COPY TO tmp2 FIELD  PARTNO, MATERNAME, QTITY
SELECT SECONDARY
USE tmp2
IF .NOT. EOF
  * Print the heading
  ?'                                                  '+DATE()
  ?'                                                  --------'
  ?'                          '+title1
  ?'                          '+under11
  ?'           '+title2
  ?'           '+under12
  ?
  STORE  1  TO seqn
  STORE  0  TO lncount
  ?'-------------------------------------------------'
  ?'|Seq#|    Partno     |   Material  name     |Qtity |'
  ?'|----|--------------|----------------- ---|------|'
  STORE  lncount+9  TO  lncount
  DO WHILE .NOT. EOF
     IF  lncount > 53
         EJECT
         STORE  0  TO  lncount
     ENDIF
     ?'|    |              |                        |      |'
     ?'|'+STR(seqn,3)
     ??'|' +partno
     ??'|' +matername
     ??'|' +STR(qtity,5)
```

```
            ??'|'
            STORE  seqn+1  TO seqn
            STORE  lncount+2  TO  lncount
            SKIP
         ENDDO  WHILE .NOT. EOF
         ?'-------------------------------------------------'
      ELSE
         ?
         ?
         ?' None material need reorder'
         ?
         ?
      ENDIF
      ?
      ?
      * SET  PRINT ON
      USE  ORDER
      GO TOP
      DELETE  NEXT  10000
      PACK
      ?
      ?
   ELSE
      ?
      ?'     ****     Incorrect  argument      ****'
      ?
      ?
   ENDIF
   ?
   ?
   ?
   ?
   ?
   ?
   ?
   ?
   ?
   ?
   STORE ' ' to continue
   @ 22,02 SAY 'Enter <y> to continue or <n> to stop -->' ,
               GET continue
   READ
   ?
ENDDO
ERASE
SET TALK ON
RETURN
```

2. Submenu2 (SUBMENU2)

```
ERASE
?
?'     ************************************************'
?'     *                                              *'
?'     *              REORDER REPORTING               *'
?'     *              -----------------               *'
?'     *                 (Subprog2)                   *'
?'     *                                              *'
?'     *     The  program  is  ready  to  run  when    *'
?'     *                                              *'
?'     *  you enter <y>.  Please, wait a few minutes !! *'
?'     *                                              *'
?'     *  Shortly a list of materials, to be needed   *'
?'     *                                              *'
?'     *  reordered will appear on the screen.        *'
?'     *                                              *'
?'     ************************************************'
?
RETURN
```

APPENDIX D

Appendix D contains Subprogram3 (SUBPROG3) and Submenu3 (SUBMENU3) with all the dependent subprograms: SUBPRG31, SUBMEN31, SUBPRG32, SUBMEN32, SUBPRG33, SUBMEN33, SUBPRG34, SUBMEN34, SUBPRG35, SUBMEN35, SUBPRG36, SUBMEN36, SUBPRG37, SUBMEN37, SUBPRG38, SUBMEN38, SUBPRG39, SUBMEN39.

1. Subprogram3 (SUBPROG3)

```
ERASE
SET TALK OFF
STORE F TO FINISH3
  DO WHILE  .NOT. FINISH3
    ERASE
    DO SUBMENU3
    STORE ' ' TO SUBMN3
    @ 25,25 SAY 'Your selection -->' GET SUBMN3 PICTURE '9
    READ
    DO  CASE
       CASE  SUBMN3 = '1'
             DO   subprg31
       CASE  SUBMN3 = '2'
             DO   subprg32
       CASE  SUBMN3 = '3'
             DO   subprg33
       CASE  SUBMN3 = '4'
             DO   subprg34
       CASE  SUBMN3 = '5'
             DO   subprg35
       CASE  SUBMN3 = '6'
             DO   subprg36
       CASE  SUBMN3 = '7'
             DO   subprg37
       CASE  SUBMN3 = '8'
             DO   subprg38
       CASE  SUBMN3 = '9'
             DO   subprg39
       CASE  SUBMN3 = '0'
             STORE  T  TO  FINISH3
    ENDCASE
  ENDDO WHILE .NOT. FINISH3
  RELEASE  FINISH3
SET TALK ON
RETURN
```

2. Submenu3 (SUBMEN3)

```
ERASE
@  1,60 SAY DATE()
@  2,10 SAY '*************************************************'
@  3,10 SAY '*                  REPORT GENERATOR             *'
@  4,10 SAY '*                  ----------------             *'
@  5,10 SAY '* code     T a s k  D e s c r i p t i o n       *'
@  6,10 SAY '* ----     -----------------------------       *'
@  7,10 SAY '*  1..  Materials received on a particular DATE  *'
@ 08,10 SAY '*  2..  Materials received in a PERIOD of TIME   *'
@ 09,10 SAY '*  3..  Materials received by a REGISTRATION #   *'
@ 10,10 SAY '*  4..  Materials given on a specific DATE or    *'
@ 11,10 SAY '*       PERIOD of TIME.                          *'
@ 12,10 SAY '*  5..  Units received a specific material in    *
@ 13,10 SAY '*       a PERIOD of TIME                         *'
@ 14,10 SAY '*  6..  Units received a specific material.      *'
@ 15,10 SAY '*  7..  Materials OWED to a specific unit.       *'
@ 16,10 SAY '*  8..  Units to which a material is owed.       *'
@ 17,10 SAY '*  9..  Spare-parts needed to repair a specific  *'
@ 18,10 SAY '*       main material.                           *'
@ 19,10 SAY '*  0..  Return to mainmenu                       *'
@ 20,10 SAY '*************************************************'
@ 22,10 SAY '      Select one of the above Task - codes '
?
?
RETURN
```

3. Subprogram31 (SUBPRG31)

```
ERASE
SET TALK OFF
DO submen31
STORE ' '  TO enterit
@ 20,03 SAY 'Enter <y> to start or <n> to stop --->',
            GET enterit
READ
DO WHILE enterit <> 'n'
  ERASE
  IF  enterit = 'y'
    STORE  '        '  TO  dt
    @ 10,12 SAY 'Enter   the   desired   DATE =======>>',
                GET dt  PICTURE '99/99/99'
    READ
    STORE 'LIST' TO title1
    STORE '====' TO underl1
    STORE 'Off all materials received on'  TO title2
    STORE '-------------------------------------' ,
          TO underl2
```

```
  USE inventory
    COPY TO tmp1 FIELD PARTNO, MATERNAME, QUANTITY,,
         REGENTERNO  FOR (regentdate = dt)
  USE tmp1
  IF .NOT. EOF
      * Print  the   heading
      ?'                                                    ',
          +DATE()
      ?'
        --------'
      ?'                              '+title1
      ?'                              '+under11
      ?'        '+title2
      ?? dt
      ?'        '+under12
      ?
      STORE  1  TO seqn
      STORE  0  TO lncount
      ?'-------------------------------------------------------
      ?'|Seq#|    Partno     |  Material name |Q'tity |Regetno|'
      ?'|----|--------------|----------------|-------|-------|'
      STORE  lncount+9  TO  lncount
      DO  WHILE  .NOT. EOF
         IF  lncount > 53
             EJECT
             STORE  0  TO  lncount
         ENDIF
         ?'|    |              |                |       |       |'
         ?'|'+STR(seqn,3)
         ??'|' +partno
         ??'|' +maternname
         ??'|' +STR(quantity,5)
         ??'|' +STR(regenterno,6)
         ??'|'
         STORE  seqn+1  TO  seqn
         STORE  lncount+2  TO  lncount
         SKIP
      ENDDO WHILE .NOT. EOF
      ?'-------------------------------------------------------
  ELSE
        ?
        ?'  None material enter this date in the unit or'
        ?'  perhaps you typed invalid form for date'
        ?
  ENDIF
ELSE
  ?
  ?'  ** incorrect letter or symbol **'
ENDIF
?
?
```

```
  ?
  ?
  ?
  STORE ' ' TO  enterit
  @ 23,07 SAY 'Enter <y> to continue or <n> to stop --->',
              GET enterit
  READ
ENDDO
ERASE
SET TALK ON
RETURN
```

4. Submenu31 (SUBMEN31)

```
ERASE
?
?'*****************************************************'
?'*                                                   *'
?'*     MATERIALS RECEIVED ON A PARTICULAR DATE       *'
?'*     ---------------------------------------       *'
?'*                    (Subprg31)                     *'
?'*                                                   *'
?'*     This program creates a list of materials      *'
?'*  received by the Ordnance Battalion on a          *'
?'*  particular DATE.                                 *'
?'*     Enter <y>, and then the DATE, and wait a      *'
?'*  few seconds.  On the screen you will see the     *'
?'*  list of materials received that DATE.            *'
?'*                                                   *'
?'*****************************************************'
?
RETURN
```

5. Subprogram32 (SUBPRG32)

```
ERASE
SET TALK OFF
DO submen32
STORE ' '  TO enterit
@ 20,07 SAY 'Enter <y> to start or <n> to stop --->' GET enterit
READ
DO WHILE enterit <> 'n'
  ERASE
  IF  enterit = 'y'
    STORE  '          '  TO  sdt
```

```
STORE  '          '  TO  edt
@ 10,12 SAY 'Enter  the  start  DATE.  FROM =====>>' ,
             GET sdt PICTURE '99/99/99'
@ 11,12 SAY '                                TO   ------>',
             GET edt PICTURE '99/99/99'
READ
STORE 'LIST' TO title1
STORE '====' TO under11
STORE 'Off all materials received from'  TO title2
STORE '-------------------------------------------------',
      TO  under12
USE inventory
  COPY TO tmp1 FIELD PARTNO, MATERNAME, QUANTITY,,
                     REGENTERNO, REGENTDATE,
       FOR (regentdate >= sdt .AND. regentdate < edt)
USE tmp1
SORT on regentdate to P1
USE P1
IF .NOT. EOF
  * Print  the   heading
  ?'                                                    '+DATE()
  ?'                                                    ---------
  ?'                           '+title1
  ?'                           '+under11
  ?'        '+title2
  ?? sdt
  ?? 'to'
  ?? edt
  ?'        '+under12
  ?
  STORE  1  TO seqn
  STORE  0  TO lncount
  ?'-------------------------------------------------------------
  ?'|Seq#|   Partno    | Material name |Qtity|Regeno|Regedt|'
  ?'|----|-------------|---------------|-----|------|------|'
  STORE  lncount+9  TO  lncount
  DO  WHILE  .NOT. EOF
    IF  lncount > 53
        EJECT
        STORE  0  TO  lncount
    ENDIF
    ?'|    |             |               |     |      |      |'
    ?'|'+STR(seqn,3)
    ??'|' +partno
    ??'|' +maternname
    ??'|' +STR(quantity,5)
    ??'|' +STR(regenterno,6)
    ??'|' +regentdate
    ??'|'
    STORE  seqn+1  TO  seqn
    STORE  lncount+2  TO  lncount
```

```
        SKIP
      ENDDO WHILE .NOT. EOF
      ?'-------------------------------------------------------'
    ELSE
          ?
          ?'  None material enter this PERIOD of TIME in '
          ?'  the Battalion'
          ?
    ENDIF
  ELSE
    ?
    ?'  ** incorrect letter or symbol **'
  ENDIF
  ?
  ?
  ?
  ?
  ?
  STORE ' ' TO  enterit
  @ 23,07 SAY 'Enter <y> to continue or <n> to stop --->',
              GET enterit
  READ
ENDDO
ERASE
SET TALK ON
RETURN
```

6. Submenu32 (SUBMEN32)

```
ERASE
?
?'*************************************************************'
?'*                                                           *'
?'*       MATERIALS RECEVEIVED IN A PERIOD OF TIME            *'
?'*       ----------------------------------------            *'
?'*                          (Subprg32)                       *'
?'*                                                           *'
?'*     The program is ready to run when you enter            *'
?'*   <y>, and then the PERIOD of TIME.                       *'
?'*     Please, wait a few seconds  !!!                       *'
?'*     A list of materials received by the Battalion         *'
?'*   in that PERIOD of TIME will appear on the screen.       *'
?'*     The materials will be sorted on Registration          *'
?'*   number.                                                 *'
?'*                                                           *'
?'*************************************************************'
?
RETURN
```

7. Subprogram33 (SUBPRG33)

```
ERASE
SET TALK OFF
DO submen33
STORE ' '  TO enterit
@ 18,07 SAY 'Enter <y> to start or <n> to stop --->' GET enterit
READ
DO WHILE enterit <> 'n'
  ERASE
  IF  enterit = 'y'
    STORE  '       '  TO  rn
    @ 10,12 SAY 'Enter the desired Registration no ===>>',
                GET rn  PICTURE '999999'
    READ
    STORE val(rn)  TO ren
    STORE 'LIST' TO title1
    STORE '====' TO under11
    STORE 'Off all materials received with'  TO title2
   STORE '-------------------------------------------------',
         TO  under12
    USE STATUS
      COPY TO tmp1 FIELD PARTNO, MATERNAME, QUANTITY,,
              REGEXDATE,REGENTERNO FOR (regenterno = ren)
    USE tmp1
    SORT on regenterno to P1
    USE P1
    IF .NOT. EOF
      * Print  the   heading
      ?'                                                    '+DATE()
      ?'                                                    --------'
      ?'                                  '+title1
      ?'                                  '+under11
      ?'        '+title2
      ?? STR(ren,6)
      ?? 'Regenterno'
      ?'        '+under12
      ?
      STORE  1  TO seqn
      STORE  0  TO lncount
      ?'-------------------------------------------------------'
      ?'|Seq#|   Partno    | Material name  |Qtit|Regeno|Regedt|'
      ?'|----|------------|----------------|----|------|------|'
      STORE  lncount+9  TO  lncount
      DO  WHILE  .NOT. EOF
        IF  lncount > 53
            EJECT
            STORE  0  TO  lncount
        ENDIF
        ?'|    |            |                |    |      |      |'
```

```
        ?'¦'+STR(seqn,3)
        ??'¦' +partno
        ??'¦' +matername
        ??'¦' +STR(quantity,5)
        ??'¦' +STR(regenterno,6)
        ??'¦' +regexdate
        ??'¦'
        STORE  seqn+1  TO  seqn
        STORE  lncount+2  TO  lncount
        SKIP
      ENDDO WHILE .NOT. EOF
      ?'-------------------------------------------------------
    ELSE
          ?
          ?'  None material enter the BATTALION with '
          ?'  this REGENTERNO'
          ?
    ENDIF
  ELSE
    ?
   "?'  ** incorrect letter or symbol **'
  ENDIF
  ?
  STORE ' ' TO  enterit
  @ 23,07 SAY 'Enter <y> to continue or <n> to stop --->'
               GET enterit
  READ
ENDDO
ERASE
SET TALK ON
RETURN
```

8. Submenu33 (SUBMEN33)

```
ERASE
?'*************************************************************'
?'*                                                           *'
?'*      MATERIALS RECEIVED BY A REGISTRATION NUMBER          *'
?'*      ------------------------------------------           *'
?'*                          (Subprg33)                       *'
?'*                                                           *'
?'*     This program gives a list of materials received       *'
?'*   by the Battalion by a specific registration number.     *'
?'*     The only thing to do is to enter <y>, and then        *'
?'*   the Regenterno, and to wait a few seconds.              *'
?'*     On the screen you will see the that list.             *'
?'*                                                           *'
?'*************************************************************'
RETURN
```

9. Subprogram34 (SUBPRG34)

```
ERASE
SET TALK OFF
DO submen34
STORE ' '  TO givenit
@ 22,07 SAY 'Enter <y> to start or <n> to stop --->' GET givenit
READ
DO WHILE givenit <> 'n'
 IF  givenit = 'y'
  ERASE
  STORE ' '  TO  d
  @ 09,10 SAY 'Type 1 for DATE or 2 for PERIOD of TIME ===>> ',
               GET d  PICTURE '9'
  READ
  STORE  val(d)  TO  dd
  IF dd = 1
   STORE '          ' TO un
   @ 12,10 SAY 'Enter   the   desired  name   of    UNIT ----> ',
               GET un
   READ
   STORE  '          '  TO  dt
   @ 15,10 SAY 'Enter    the    desired    DATE  ==========> ',
               GET dt  PICTURE '99/99/99'
   READ
   STORE 'LIST' TO title1
   STORE '====' TO under11
   STORE 'Off all materials given to'  TO title2
   STORE '-----------------------------------------------',
         TO  under12
   USE materialy
    COPY TO tmp1 FIELD PARTNO,MATERNAME,QTITY,REGEXNO,
          FOR (unitname = un .AND. regexdate = dt)
   USE tmp1
    IF .NOT. EOF
     ?
     * Print  the   heading
     ?'                                                    '+DATE()
     ?'                                                   ----------'
     ?'                          '+title1
     ?'                          '+under11
     ?'    '+title2
     ?? un
     ?? 'on'
     ?? dt
     ?'    '+under12
     ?
     ?
     STORE  1  TO seqn
     STORE  0  TO incount
```

```
    ?'-------------------------------------------------------'
    ?'|Seq#|    Partno    |  Material name    |Qtity |Regexno|'
    ?'|----|--------------|-------------------|------|-------|'
    STORE  lncount+9  TO  lncount
    DO  WHILE  .NOT. EOF
     IF  lncount > 53
         EJECT
         STORE  0  TO  lncount
     ENDIF
     ?'|    |              |                   |      |       |'
     ?'|'+STR(seqn,3)
     ??'|' +partno
     ??'|' +matername
     ??'|' +STR(qtity,5)
     ??'|' +STR(regexno,6)
     ??'|'
     STORE  seqn+1  TO  seqn
     STORE  lncount+2  TO  lncount
     SKIP
    ENDDO WHILE .NOT. EOF
    ?'-------------------------------------------------------'
   ELSE
    ?
    ?'  None material was given this DATE to that UNIT'
    ?'  Check the typed DATE and UNIT name'
   ENDIF
ELSE
 IF dd = 2
  STORE '          ' TO un
  @ 12,10 SAY 'Enter   the   desired   name   of   UNIT ---->',
              GET un
  READ
  STORE  '          '  TO  sdt
  STORE  '          '  TO  edt
  @ 15,10 SAY 'Enter  the  start  DATE.  FROM =====>>' ,
              GET sdt  PICTURE '99/99/99'
  @ 16,10 SAY '                              TO   ------>' ,
              GET edt  PICTURE '99/99/99'
  READ
  STORE 'LIST' TO title1
  STORE '====' TO under11
  STORE 'Off all materials given to'  TO title2
  STORE '-------------------------------------------------' ,
        TO under12
  USE material
    COPY TO tmp1 FIELD PARTNO, MATERNAME, QTITY, REGEXNO,,
            REGEXDATE FOR (unitname = un .AND. ,
            (regexdate >= sdt .AND. regexdate <= edt))
  USE tmp1
  SORT on regexdate to P1
  USE P1
```

```
  IF .NOT. EOF
    * Print  the   heading
    ?'                                                          +DATE()
    ?'                                                          ----------'
    ?'                                        '+title1
    ?'                                        '+under11
    ?'  '+title2
    ?? un
    ?? 'from'
    ?? sdt
    ?? 'to'
    ?? edt
    ?'  '+under12
    ?
    STORE  1  TO seqn
    STORE  0  TO lncount
    ?'-------------------------------------------------------------'
    ?'|Seq#|   Partno    | Material name  |Qtity|Regen|Regedt|'
    ?'|----|-------------|----------------|-----|-----|------|'
    STORE  lncount+9  TO  lncount
    DO  WHILE  .NOT. EOF
      IF  lncount > 53
          EJECT
          STORE  0  TO  lncount
      ENDIF
      ?'|    |             |                |     |     |      |'
      ?'|'+STR(seqn,3)
      ??'|' +partno
      ??'|' +matername
      ??'|' +STR(qtity,5)
      ??'|' +STR(regexno,6)
      ??'|' +regexdate
      ??'|'
      STORE  seqn+1  TO  seqn
      STORE  lncount+2  TO  lncount
      SKIP
    ENDDO WHILE .NOT. EOF
    ?'-------------------------------------------------------------'
  ELSE
    ?
    ?'  None material was given to this UNIT that '
    ?'  PERIOD of TIME'
    ?
  ENDIF
 ELSE
  ?
  ?'Error you typed incorrect number'
  ?'Note correct numbers are  1  or  2'
  ?
 ENDIF
ENDIF
```

```
 ELSE
  ?
  ?'  ** incorrect letter or symbol **'
  ?
 ENDIF
  ?
  ?
  ?
  ?
  ?
  ?
  ?
  ?
  STORE ' ' TO  givenit
  @ 23,07 SAY 'Enter <y> to continue or <n> to stop --->',
              GET givenit
  READ
ENDDO
ERASE
SET TALK ON
RETURN
```

10. Submenu34 (SUBMEN34)

```
ERASE
?
?
?'*************************************************************'
?'*                                                           *'
?'*    MATERIALS GIVEN ON A DATE OR IN A PERIOD OF TIME       *'
?'*    ------------------------------------------------       *'
?'*                        (Subprg34)                         *'
?'*                                                           *'
?'*                                                           *'
?'*     The program gives lists of materials given by the     *'
?'*  Battalion to  Units on a specific DATE or in a           *'
?'*  PERIOD of TIME.                                          *'
?'*     Enter <y> and then select option 1 for DATE or        *'
?'*  option 2 for PERIOD of TIME.                             *'
?'*     Then the program is ready to run when you type        *'
?'*  the specific DATE or PERIOD of TIME.                     *'
?'*     Wait a few seconds.  On the screen you will the       *'
?'*  list.                                                    *'
?'*                                                           *'
?'*************************************************************'
?
RETURN
```

11. Subprogram35 (SUBPRG35)

```
ERASE
SET TALK OFF
DO submen35
STORE ' '  TO givenit
@ 18,07 SAY 'Enter <y> to start or <n> to stop --->' GET givenit
READ
DO WHILE givenit <> 'n'
  ERASE
  IF  givenit = 'y'
    STORE '                '  TO  pn
    @ 07,10 SAY 'Enter the Part Number of the material---->' ,
                GET pn  PICTURE '9999-999-9999'
    READ
    STORE  '          '  TO  sdt
    STORE  '          '  TO  edt
    @ 10,10 SAY 'Enter  the  start  DATE.  FROM ========>>' ,
                GET sdt    PICTURE '99/99/99'
   @ 11,10 SAY '                                TO   ----------->" ,
                GET edt   PICTURE '99/99/99'
    READ
    STORE 'LIST' TO title1
    STORE '====' TO under11
    STORE 'Off UNITS received '' TO title2
    STORE '-----------------------------------'  TO  under12
    STORE 'From'  TO  title3
    STORE '-------------------------'  TO under13
    USE material
     COPY TO tmp1 FIELD PARTNO, MATERNAME, QTITY, REGEXNO,,
                  REGEXDATE, UNITNAME,
        FOR (partno = pn .AND. (regexdate >= sdt .AND. .
             regexdate <= edt))
    USE tmp1
    SORT on regexdate to P1
    USE P1
    IF .NOT. EOF
      * Print  the   heading
      ?'                                                    '+DATE()
      ?'                                                   ---------'
      ?'                              '+title1
      ?'                              '+under11
      ?'          '+title2
      ?? pn
      ?? '  '
      ?'          '+under12
      ?'               '+title3
      ?? sdt
      ?? 'to'
      ?? edt
```

```
      ?'                '+under13
      ?
      STORE  1  TO seqn
      STORE  0  TO lncount
      ?'-------------------------------------------------------'
      ?'|Seq#|      Unit  name      |Qtity |Regexno|Regexdate|'
      ?'|----|--------------------|------|-------|---------|'
      STORE  lncount+9  TO  lncount
      DO  WHILE  .NOT. EOF
        IF  lncount > 53
          EJECT
          STORE  0  TO  lncount
        ENDIF
        ?'|    |                    |      |       |         |'
        ?'|'+STR(seqn,3)
        ??'|' +unitname
        ??'|' +STR(qtity,5)
        ??'|' +STR(regexno,6)
        ??'|' +regexdate
        ??'|'
        STORE  seqn+1  TO  seqn
        STORE  lncount+2  TO  lncount
        SKIP
      ENDDO WHILE .NOT. EOF
      ?'-------------------------------------------------------'
    ELSE
          ?
          ?'  None unit of the Division received this material
          ?'   during given Period of Time'
          ?
    ENDIF
  ELSE
    ?
    ?'  ** incorrect letter or symbol **'
  ENDIF
  ?
  ?
  ?
  ?
  ?
  ?
  ?
  ?
  STORE ' ' TO  givenit
  @ 23,07 SAY 'Enter <y> to continue or <n> to stop'--->',
              GET givenit
  READ
ENDDO
ERASE
SET TALK ON
RETURN
```

12. Submenu35 (SUBMEN35)

```
ERASE
?
?'*************************************************************'
?'*                                                           *'
?'*    UNITS RECEIVED A MATERIAL IN A PERIOD OF TIME          *'
?'*    ---------------------------------------------          *'
?'*                        (Subprg35)                         *'
?'*                                                           *'
?'*     The program is ready to run when you enter            *'
?'*  <y>, and then the part number of the material and        *'
?'*  the PERIOD of TIME.                                      *'
?'*     Please, wait a few secods !!!                         *'
?'*     Sortly, a list of units, received the specific        *'
?'*  material in that period of time will appear on           *'
?'*  the screen.                                              *'
?'*                                                           *'
?'*************************************************************'
?
RETURN
```

13. Subprogram36 (SUBPRG36)

```
ERASE
SET TALK OFF
DO submen36
STORE ' '  TO givenit
@ 18,07 SAY 'Enter <y> to start or <n> to stop --->' GET givenit
READ
DO WHILE givenit <> 'n'
  ERASE
  IF  givenit = 'y'
    STORE '                ' TO  pn
    @ 07,10 SAY 'Enter the Part Number of the material--->' ,
                GET pn  PICTURE '9999-999-9999'
    READ
    STORE 'LIST' TO title1
    STORE '====' TO under11
    STORE 'Off UNITS received ' TO title2
    STORE '----------------------------------'  TO under12
    USE material
     COPY TO tmp1 FIELD PARTNO, MATERNAME, QTITY, REGEXNO,,
                  REGEXDATE, UNITNAME  FOR (partno = pn)
    USE tmp1
    SORT on regexdate to P1
    USE P1
```

```
    IF .NOT. EOF
      * Print  the   heading
      ?'                                                        +Date
      ?'                                                        ---------'
      ?'                                '+title1
      ?'                                '+under11
      ?'          '+title2
      ?? pn
      ?? '''
      ?'          '+under12
      ?
      STORE  1  TO seqn
      STORE  0  TO lncount
      ?'-------------------------------------------------------------
      ?'|Seq#|      Unit  name      |Qtity |Regexno|Regexdate|'
      ?'|----|---------------------|------|-------|---------|'
      STORE  lncount+9  TO  lncount
      DO  WHILE  .NOT. EOF
        IF  lncount > 53
            EJECT
            STORE  0  TO  lncount
        ENDIF
        ?'|    |                     |      |       |         |'
        ?'|'+STR(seqn,3)
        ??'|' +unitname
        ??'|' +STR(qtity,5)
        ??'|' +STR(regexno,6)
        ??'|' +regexdate
        ??'|'
        STORE  seqn+1  TO  seqn
        STORE  lncount+2  TO  lncount
        SKIP
      ENDDO WHILE .NOT. EOF
      ?'-------------------------------------------------------------
    ELSE
          ?
          ?'  None unit of the Division received this material'
          ?
    ENDIF
  ELSE
    ?
    ?'  ** incorrect letter or symbol **'
  ENDIF
  STORE ' ' TO  givenit
  @ 23,07 SAY 'Enter <y> to continue or <n> to stop ---->' ,
              GET givenit
  READ
ENDDO
ERASE
SET TALK ON
RETURN
```

14. Submenu36 (SUBMEN36)

```
ERASE
?
?'*************************************************************'
?'*                                                           *'
?'*       LIST OF UNITS RECEIVED A SPECIFIC MATERIAL.         *'
?'*       ------------------------------------------          *'
?'*                          (Subprg36)                       *'
?'*                                                           *'
?'*      The program is ready to run when you enter           *'
?'*  <y>, and then the part number of the specific            *'
?'*  material.                                                *'
?'*      Please, wait a few seconds  !!!                      *'
?'*      A list of all units received that material           *'
?'*  will appear on the screen.                               *'
?'*                                                           *'
?'*************************************************************'
?
RETURN
```

15. Subprogram37 (SUBPRG37)

```
ERASE
SET TALK OFF
DO SUBMEN37
STORE ' ' to owedit
@ 18,04 SAY 'Enter <y> to continue or <n> to stop -->' ,
            GET owedit
READ
DO WHILE owedit <> 'n'
  ERASE
  IF  owedit = 'y'
    STORE  '                          ' to unitnam
    @ 10,10 SAY 'Enter the unit name ==>>' GET unitnam
    READ
    USE owed
      COPY TO tmp1 FIELD PARTNO, MATERNAME, QTITY, DUEDATE,,
           REGEXNO  FOR (unitname = unitnam)
      USE tmp1
      IF .NOT. EOF
          ?
          ?'                                L I S T '
          ?'                                -------'
          ?'          Off all materials owed to  '
          ?? unitnam
          ?'          ------------------------------------------'
```

```
            ??'Seq#'
            ??'       Partno    '
            ??'    Material name    '
            ??'         Q'
            ??' Duedate '
            ??'Regexno'
            ?'----  ---------- --------------  ---- ------  ------'
            LIST
            ?
        ELSE
            ?
            ?'  None material owed to this unit  or perhaps you
            ?'  typed an invalid form for unitname'
        ENDIF
    ELSE
        ?
        ?'  ** incorrect letter or symbol **'
    ENDIF
    ?
    ?
    ?'
    STORE ' ' to owedit
    @ 22,10 SAY 'Enter <y> to continue or <n> to stop -->' ,
                GET owedit
    READ
ENDDO
ERASE
SET TALK ON
RETURN
```

16. Submen37 (SUBMEN37)

```
ERASE
?'*******************************************************'
?'*                                                     *'
?'*       MATERIALS OWED TO A SPECIFIC UNIT             *'
?'*       ---------------------------------             *'
?'*                    (Subprg37)                       *'
?'*                                                     *'
?'*     This program creates a list of materials        *'
?'*  owed by the Ordnance Battalion to a                *'
?'*  specific UNIT.                                     *'
?'*     Enter <y>, and then the name of of unit.        *'
?'*     Please, wait a few seconds !!!                  *'
?'*     On the screen you will see the list of          *'
?'*  materials owed to that unit.                       *'
?'*                                                     *'
?'*******************************************************'
RETURN
```

17. Subprogram38 (SUBPRG38)

```
ERASE
SET TALK OFF
DO submen38
STORE ' ' to owedit
@ 18,05 SAY 'Enter <y> to continue or <n> to stop -->',
            GET owedit
READ
DO WHILE owedit <> 'n'
  ERASE
  IF  owedit = 'y'
    STORE '               ' to partnum
    STORE '                        ' to maternam
    @ 10,10 SAY 'Enter part number of material -->',
                GET partnum PICTURE '9999-999-9999'
    @ 11,10 SAY 'Enter the name of material ====>>',
                GET maternam
    READ
   USE owed
      COPY TO tmp1 FIELD UNITNAME,QTITY,DUEDATE,REGEXNO,
           FOR (partno = partnum .AND. matername = maternam)
      USE tmp1
      IF .NOT. EOF
          ?'                          L I S T '
          ?'                          -------'
          ?'Off all units to which'
          ??'  '
          ?? maternam
          ??'is owed'
          ?'-------------------------------------------------
          ?
          ?
          ??'Seq#'
          ??'        Unit  name     '
          ??'      Q'
          ??' Duedate '
          ??'Regexno'
          ?'-----  ----------------  ------ --------  -------
          LIST
          ?
      ELSE
          ?
          ?'  None unit is owed this material  or perhaps'
          ?'  you typed invalid part number or name'
          ?
      ENDIF
  ELSE
     ?
     ?'  ** incorrect letter or symbol **'
```

```
   ENDIF
   ?
   ?
   ?
   ?
   ?
   ?
   ?
   ?
   ?
   ?
   STORE ' ' to owedit
   @ 22,10 SAY 'Enter <y> to continue or <n> to stop --> ',
                GET owedit
   READ
   ?
ENDDO
CLEAR
SET TALK ON
RETURN
```

18. Submenu38 (SUBMEN38)

```
ERASE
?
?'*************************************************************'
?'*                                                           *'
?'*                                                           *'
?'*      LIST OF UNITS TO WHICH A MATERIAL IS OWED            *'
?'*      ----------------------------------------             *'
?'*                      (Subprg38)                           *'
?'*                                                           *'
?'*                                                           *'
?'*     The program is ready to run when you enter            *'
?'*  <y>, and then the part number of the specific            *'
?'*  material, and its name.                                  *'
?'*     Please, wait a few seconds  !!!                       *'
?'*     A list of all units to which that material is         *'
?'*  owed will appear on the screen.                          *'
?'*                                                           *'
?'*************************************************************'
?
?
RETURN
```

19. Subprogram39 (SUBPRG39)

```
ERASE
SET TALK OFF
DO submen39
STORE ' ' to spareit
@ 18,07 SAY 'Enter <y> to continue or <n> to stop -->' ,
            GET spareit
READ
DO WHILE spareit <> 'n'
  ERASE
  IF spareit = 'y'
   STORE  '                    '  TO mn
   @ 10,12 SAY 'Enter the name of material ==>>' GET mn
   READ
   USE INVENTORY
      COPY TO tmp1 FIELD PARTNO, MATERNAME, QUANTITY, OWED,,
                   WAITED  FOR (mainmater = mn)
      USE tmp1
      IF .NOT. EOF
          ?
          ?'                                L I S T '
          ?'                                -------'
          ?'          Off all spare parts used from '
          ?? mn
          ?'          ----------------------------------------
          ?
          ?
          ??'Seq#'
          ??'       Partno    '
          ??'   Material name    '
          ??'         Q'
          ??' owed '
          ??'waited'
          ?'-----  ---------- ------------  -----  ----  -------'
          LIST
          ?
      ELSE
          ?
          ?' None from the existing in the unit spare parts
          ?' can be used for repair of this main material,'
          ?' Or you typed invalid main material name'
      ENDIF
    ELSE
      ?
       ?'  ** incorrect letter or symbol **'
ENDIF
?
?
?
```

```
?
STORE ' ' TO spareit
@ 22,10 SAY 'Enter <y> to continue or <n> to stop -->' ,
            GET spareit
READ
ENDDO
CLEAR
SET TALK ON
RETURN
```

20. Submenu39 (SUBMEN39)

```
ERASE
?
?'*************************************************************
?'*                                                           *
?'*     LIST OF SPARE-PARTS NEEDED TO REPAIR A MATERIAL       *
?'*     ---------------------------------------------         *
?'*                     (S u b p r g 3 9)                     *
?'*                                                           *
?'*      The program is ready to run as soon as you enter     *
?'*  <y>, and then the name of the material.                  *
?'*      Wait a few seconds, please.                          *
?'*      On the screen you will see the list of spare-parts,  *
?'*  needed to repair the specific main material.             *
?'*                                                           *
?'*************************************************************
?
RETURN
```

APPENDIX E

Appendix E contains the subprogram4 (SUBPROG4) and submenu4 (SUBMENU4) and the dependent subprograms: SUBPRG41, SUBMEN41, SUBPRG42, and SUBMEN42.

1. Subprogram4 (SUBPROG4)

```
ERASE
SET TALK OFF
STORE  F  TO FINISH4
DO WHILE .NOT. FINISH4
   DO submenu4
   STORE ' ' TO choise4
   @ 20,20 SAY 'Your selection --->' GET choise4
   READ
   ?
   DO WHILE .NOT. choise4 $ '012'
      @ 20,42 SAY '<<=== Invalid Choice.Re-enter'
      STORE ' ' TO choise1
      @ 20,20 SAY 'Your selection --->' GET choise4
      READ
   ENDDO
   ?
   DO  CASE
       CASE  choise4 = '1'
             DO   subprg41
       CASE  choise4 = '2'
             DO   subprg42
       CASE  choise4 = '0'
             STORE  T  TO  FINISH4
   ENDCASE
?
ENDDO
RELEASE FINISH4
SET TALK ON
RETURN
```

2. Submenu4 (SUBMENU4)

```
ERASE
@  1,10 SAY DATE()
@  2,10 SAY '*************************************************'
@  3,10 SAY '*                                               *'
@  4,10 SAY '*                   LOOKS--UPS                  *'
@  5,10 SAY '*                   ----------                  *'
@  6,10 SAY '*                    (Submenu4)                 *'
@  7,10 SAY '*                                               *'
@  8,10 SAY '* code      T a s k   D e s c r i p t i o n     *'
@ 09,10 SAY '* ----      ---------------------------------    *'
@ 10,10 SAY '*                                               *'
@ 11,10 SAY '*  1..  Decision to give a material             *'
@ 12,10 SAY '*                                               *'
@ 13,10 SAY '*  2..  Information status of material          *'
@ 14,10 SAY '*                                               *'
@ 15,10 SAY '*  0..  Return to mainmenu                      *'
@ 16,10 SAY '*                                               *'
@ 17,10 SAY '*************************************************'
@ 18,20 SAY 'Select one of the above codes'
RETURN
```

3. Subprogram41 (SUBPRG41)

```
ERASE
SET TALK OFF
DO SUBMEN41
STORE ' ' TO continue
@ 20,08 SAY 'Enter <y> to run or <n> to stop ==>>' GET continue
READ
DO WHILE continue <> 'n'
  ERASE
  IF continue = 'y'
    STORE '               ' TO pn
    STORE '     '           TO qt
    @  6,12 SAY 'Enter part number   ======>>' GET pn,
                Picture '9999-999-9999'
    @  7,12 SAY 'Enter Requested Quantity -->' GET qt,
                Picture '99999'
    READ
    @ 09,07 SAY 'The program is running.,
                Please wait for the results.'
    STORE  0       TO  Q
    STORE  0       TO  PQ
    STORE  0       TO  OQ
```

```
STORE val(qt)  TO  Q
use inventory
LOCATE FOR partno = pn
IF EOF
  @ 11,12 SAY 'The requested material does not exist ,
              in Battalion'
ELSE
   IF quantity - Q > min
     @ 11,12 SAY 'You can give the requested material'
     @ 13,12 SAY 'Existing  Quantity ='
     ?? quantity
   ELSE
     IF quantity > min
       STORE  quantity - min  TO PQ
       @ 11,12 SAY 'You can give the following quantity'
       ?? PQ
     ELSE
       STORE '      '  TO QQ
       STORE '      '  TO M
       STORE '      '  TO W
       STORE '      '  TO O
       STORE quantity TO QQ
       STORE min      TO M
       STORE waited   TO W
       STORE owed     TO O
       use owed
          COPY TO tmp FIELD QTITY FOR (partno = pn)
       use tmp
       DO WHILE .NOT. EOF
          STORE QQ + QTITY TO QQ
          SKIP
       ENDDO
       @ 11,12 SAY 'I'm sorry you  can  not  give any,
                    quantity'
       @ 15,12 SAY 'Go to your  BOSS  to decide  what,
                    will do'
       @ 16,12 SAY 'ATTENTION. You need the following,
                    information'
       @ 18,12 SAY 'QUANTITY  existing        ='
       ?? QQ
       @ 19,12 SAY 'MINIMUM  quantity         ='
       ?? M
       @ 20,12 SAY 'WAITED   quantity         ='
       ?? W
       @ 21,12 SAY 'OWED qtity to  Battalion  ='
       ?? O
       @ 22,12 SAY 'Owed qtity to other units ='
       ?? QQ
     ENDIF
   ENDIF
ENDIF
```

```
  ELSE
    ?
    ?'   **  Incorrect  argument  **
    ?
  ENDIF
  ?
  ?
  ?
  ?
  ?
  ?
  ?
  ?
  ?
  ?
  ?
  STORE ' ' TO continue
  @ 24,12 SAY 'Enter <y> to continue or <n> to stop ==>  .
              GET continue
  READ
ENDDO
ERASE
SET TALK ON
RETURN
```

4. Submenu41 (SUBMEN41)

```
ERASE
?
?'    ************************************************
?'    *                                              *
?'    *          DECISION TO GIVE A MATERIAL         *
?'    *          ---------------------------         *
?'    *                  (Subprog41)                 *
?'    *                                              *
?'    *     The  program  is  ready  to  run  when   *
?'    *  you enter <y>, and then the part number of  *
?'    *  the material and the requested quantity.    *
?'    *                                              *
?'    *     You have to wait only a few seconds  !!! *
?'    *                                              *
?'    *     Shortly, on the screen you will see  a   *
?'    *  message what exactly to do.                 *
?'    *                                              *
?'    ************************************************
?
RETURN
```

5. Subprogram42 (SUBPRG42)

```
ERASE
SET TALK OFF
DO SUBMEN42
STORE ' ' TO continue
@ 20,10 SAY 'Enter <y> to run or <n> to stop ==>>' ,
            GET continue
READ
DO WHILE continue <> 'n'
  ERASE
  IF continue = 'y'
    STORE '              ' TO pn
    @  4,12 SAY 'Enter part number   ======>>' GET pn ,
                Picture '9999-999-9999'
    READ
    @  6,07 SAY 'The program is running. ,
                Please wait for the results'
    use inventory
   LOCATE FOR partno = pn
    IF EOF
      @  8,08 SAY 'The requested material does not exist ,
                  in Battalion'
    ELSE
           STORE   0       TO OQ
           STORE   0       TO GQ
           STORE   0       TO RQ
           STORE '      '  TO Q
           STORE '      '  TO MX
           STORE '      '  TO MN
           STORE '      '  TO W
           STORE '      '  TO O
           STORE quantity TO Q
           STORE max      TO MX
           STORE min      TO MN
           STORE waited   TO W
           STORE owed     TO O
           use owed
              COPY TO tmp1 FIELD QTITY FOR (partno = pn)
           use tmp1
           DO WHILE .NOT. EOF
              STORE OQ + QTITY TO OQ
              SKIP
           ENDDO
           use material
              COPY TO tmp2 FIELD QTITY FOR (partno = pn)
           use tmp2
           DO WHILE .NOT. EOF
              STORE GQ + QTITY TO GQ
              SKIP
```

```
            ENDDO
            use status
               COPY TO tmp3 FIELD QUANTITY FOR (partno = pn)
            use tmp3
            DO WHILE .NOT. EOF
               STORE  RQ + quantity TO RQ
               SKIP
            ENDDO
            @  9,12 SAY 'QUANTITY  existing        ='
            ?? Q
            @ 10,12 SAY 'MAXIMUM  quantity         ='
            ?? MX
            @ 11,12 SAY 'MINIMUM  quantity         ='
            ?? MN
            @ 12,12 SAY 'WAITED   quantity         ='
            ?? W
            @ 13,12 SAY 'OWED qtity to  Battalion  ='
            ?? O
            @ 15,12 SAY 'Owed qtity to other units ='
            ?? OO
            @ 17,12 SAY 'Given qtity this year     ='
            ?? GQ
            @ 19,12 SAY 'Received qtity this year  ='
            ?? RQ
            ?
      ENDIF
    ELSE
      ?
      ?'  **  Incorrect  argument  **  '
      ?
    ENDIF
    ?
    ?
    ?
    ?
    ?
    ?
    ?
    ?
    ?
    ?
    ?
    ?
    STORE ' ' TO continue
    @ 22,12 SAY 'Enter <y> to continue or <n> to stop ==>',
                GET continue
    READ
  ENDDO
  ERASE
  SET TALK ON
  RETURN
```

6. Submenu42 (SUBMEN42)

```
ERASE
?
?'     ************************************************'
?'     *                                              *'
?'     *         INFORMATION STATUS OF A MATERIAL     *'
?'     *         --------------------------------     *'
?'     *                   (Subprog42)                *'
?'     *                                              *'
?'     *      The  program  is  ready  to  run  when  *'
?'     *   you enter <y>, and then the part number of *'
?'     *   the material.                              *'
?'     *                                              *'
?'     *      Please, wait a few seconds !!!          *'
?'     *                                              *'
?'     *      Shortly, a list of all the information  *'
?'     *   about the material will appear on the screen. *'
?'     *                                              *'
?'     ************************************************
?
RETURN
```

LIST OF REFERENCES

1. Kroenke M. David, Database Processing, Second Edition, Science Research Associates, Inc. 1983.

2. Ullman D. Jeffrey, Principles of Database Systems, Computer Science Press, Rockville, Maryland, 1985.

3. Senn A. James, Analysis and Design of Information Systems, McGraw-Hill Book Company, 1984.

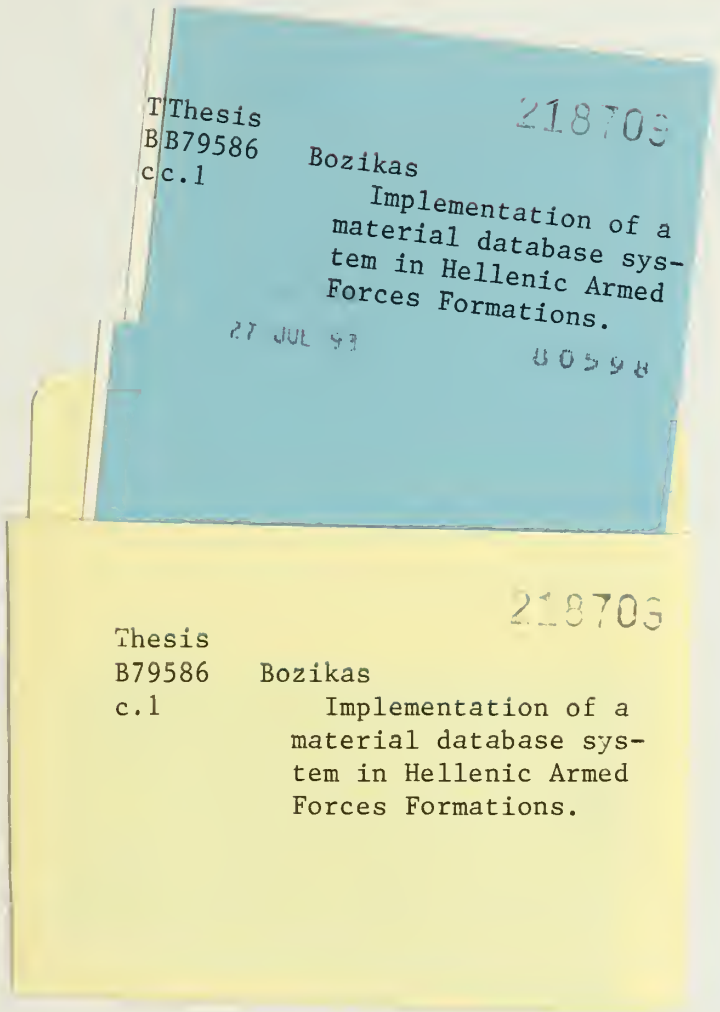
4. Freedman Alan, dBASE II for the First-Time User, The Computer Language Company, Inc., 1984.

5. Robbins Judd & Braly Ken, Expert dBASE, An Advance Textbook for dBASE Programmers, Computer Options Publications Berkley, California, 1985.

6. Castro Luis, Hanson Jay and Retting Tom, Advanced Programmer's GUIDE, Featuring dBASE III and dBASE II Ashton-Tate, 1985.

INITIAL DISTRIBUTION LIST

		No Copies
1.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22304-6145	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, California 93943-5000	2
3.	Computer Technology Curricular Office Code 37 Naval Postgraduate School Monteray, California 93943-5100	1
4.	CDR L. Rawlinson, Code 52Rv Department of Computer Science Naval Postgraduate School Monterey, California 93943-5100	1
5.	CDR Gary S. Baker, Department of Computer Science Naval Postgraduate School Monterey, California 93943-5100	1
6.	LTC Bozikas Panagiotis Hellenic Army Genaral Staff Administration of Ordnance Corp Stratopedo Papagou Holargos Athens Greece	20



218703
Thesis
B79586
c.1
Bozikas
Implementation of a material database system in Hellenic Armed Forces Formations.